ENTERPRISE ASPECTS

OF

OPEN DISTRIBUTED SYSTEMS

by

Zoran Milosevic, BSEE(Hons), MSc

A thesis submitted to
The Department of Computer Science
The University of Queensland

for the degree of

DOCTOR OF PHILOSOPHY

October 1995

Declaration

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. I hereby declare that I have not submitted this material either in whole or in part, for a degree at this or any other university.

Zoran Milosevic

September 1995

Abstract

This thesis explores new enterprise characteristics and requirements of emerging computing and telecommunications systems: the economic and business aspects of open distributed systems (ODSs).

We analyse the interplay of technological and commercial trends pertinent to ODSs and identify those enterprise factors which attract special attention from the end-user, management and technical communities involved. Relevant paradigms from economics and business are selected; it is then illustrated how they can be applied to related problems in ODSs.

Two specific concepts are investigated *i)* the enterprise notion of Quality of Service (QoS) and *ii)* new types of uncertainty inherent in a large class of services in ODSs. The aim of the former is to develop a generic, user oriented and service independent methodology which will facilitate the description and measuring of QoS in ODSs. We present a methodology which is based on the use of a specific economic theory and relevant results from market research and psychology. The objective of the latter is to address possible economic inefficiency due to uncertainty which can arise from the properties inherent in ODSs: participating parties have different characteristics, objectives and requirements, they can belong to different geographical, organisational or other domains, and yet share computing and communication resources. We address this problem from both a theoretical and a practical angle. The theoretical enquiry emanates from economic agency theory, which can be used to design efficient contracts in the presence of uncertainty. The practical approach is based on business and legal ways of handling the uncertainty associated with the interactions of economic actors in the real world: namely, the use of contracts. We derive a business contract architecture, which can facilitate business dealings in ODSs. It is based on notions of contracts drawn from economic, legal and business perspectives.

Following these theoretical and architectural researches, we present a partial implementation of the business contract architecture to demonstrate the feasibility of our approach.

Acknowledgments

I would like to sincerely thank my supervisor Prof. Andrew Lister for his expert supervision, technical advice, insightful discussions and friendly encouragement during the process of working on this thesis. I am fortunate to have had the honour and pleasure of working with Prof. Lister and I am indebted to him for his superior guidance which has enabled me to grow in thinking about problems and to write clearly about their solutions. It has indeed been an enlightening and indelible experience to have had Prof. Lister as a supervisor.

Many people have contributed to the ideas in this dissertation. I am grateful to Prof. Mirion Bearman for her assistance in further refining my initial thoughts; to Prof. Bezalel Gavish for his valuable advice regarding Agency Theory; to Prof. Lorne Mason for transferring his passion for Game Theory to me; to Dr. Ming Lai for his perceptive suggestions regarding Quality of Service issues; to Dr. Kerry Raymond, Dr. Andy Bond, Andrew Berry and David Arnold for their support in my efforts to incorporate business contract concepts into an open distributed systems architecture; to Michael Phillips for the many hours we spent discussing the interplay between performance and economic issues in open distributed systems; to Dr. Brian Hicks, Dr. Jadviga Indulska, Dr. Roger Duke, and Prof. Maria Orlowska for their willingness to discuss many questions which have arisen in the course of this research.

Australian Postgraduate Research Award has been my principal financial support. In addition, this research has been partially funded by the Cooperative Research Centres Program through the department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

It has also been privilege to work and become friends with Jean-Luc Thibault, Prof. Richard Harris, Prof. Les Berry, Prof. Zheng da Wu, and have fruitful discussions with Prof. Peter Linington, Dr. Art Gaylord, Dr. David Igullden, Peter Richardson, the late Steve Leask, Dr. Anthony Bloesch, Dr. Leith Campbell and Dr. Bob Warfield. I also thank Rob

Cook for his support and understanding and my fellow graduate students Dr. Tony Gedge, Guoping Zhang and Jason Richards for filling these days with fun, coffee, chocolate and good times.

I am eternally grateful to my parents Danica and Dr. Dragoljub Milosevic for their love, understanding and sacrifices and to my brother Dusan for his support, tolerance and humour over past years.

Last, but not least, I warmly thank Maree for always being ready to help in these years of dilemmas and uncertainty. She has been a true friend and I am grateful to her for this.

To My Parents

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 The problem domain

This thesis focuses on new enterprise related aspects of modern computer systems, in particular the economic and business aspects of emerging *open* distributed systems. Our view of openness encompasses both technical aspects (e.g. interworking of different technologies) and market aspects (e.g. open access to information services publicly offered and interworking across organisational boundaries). By 'enterprise-related', we mean those non-technological aspects which include a wide range of policy issues, such as [82]:

- who uses which information and for what purpose
- what obligations and liabilities have to be respected, e.g. sensitivity of information, charging model, regulatory requirements
- the notions of value, policy, boundary, contract negotiation and sanctions, required to model business processes behind services.

More specifically, the thesis enquires into the new, economic related aspects of interactions between users and providers in an open distributed system (ODS), the roles which Quality

of Service plays in these interactions and an architectural framework required to support them. The problem domain of the thesis belongs to the enterprise viewpoint of the Reference Model for Open Distributed Processing (RM-ODP). This is a joint ISO/ITU Standard, which provides a framework for specifying large scale, heterogeneous distributed systems [82].

1.2 Motivation

The motivation for this research stems from the facts that *i)* some new characteristics of ODSs are becoming increasingly user-oriented rather than technology-determined and *ii)* within an ODS a wide range of users with different requirements, characteristics and objectives, and using different services, can interact. In this sense, the thesis parallels changes within the scope of the information technology (IT) problem domain. Namely, the concern of the IT community has traditionally been with technological problems such as design of computer platforms, software and computer systems. It is increasingly being recognised that this concern should be extended towards a variety of non-technological requirements and even overlapped with the scope of social sciences such as psychology, sociology and economics, where appropriate.

This transition from the ‘technology push’ towards ‘user pull’, which is evident in the research, commercial and standardisation arenas is made possible by the enhanced capabilities and reduced cost of new technologies. These characteristics have enabled more user-oriented solutions at all levels of computer systems complexity, more efficient intra and inter-organisational operations, penetration of new services into the areas of residential and small business customers, as well as a more technology aware, educated and demanding community.

The recent trend of blurring boundaries between previously separate industries, computing and communications, has augmented the significance of enterprise aspects. Open distributed systems, which are at the heart of this trend, are probably the best context in which new enterprise factors can presently be studied. Although these systems currently face a number of technical challenges such as implementing distribution transparencies

(e.g. location, access, migration, replication) and integrating appropriate management solutions, the importance of enterprise issues must not be overlooked. On the contrary, an enterprise analysis should be a starting point as it may lead for example, to the identification of those distribution transparencies which need to be implemented for a particular ODS to meet its design targets.

The exploration of *economic* related enterprise aspects of ODSs has a twofold motivation.

First, we contemplate the idea that economics can be used to clarify the notion of Quality of Service (QoS) and provide guidelines for deriving a QoS metrics framework. QoS is a term increasingly being used not only in telecommunications but also in computing, although there is still a lack of a clear definition of its meaning. The ambiguity comes partly from the fact that QoS has long been regarded as an engineering variable: a network performance parameter, defined by telecommunication companies. However, owing to many new factors such as the proliferation of new services, increased competition, and user-driven QoS requirements, this view of QoS needs to be changed. Indeed, QoS needs to be defined in a service-independent manner and be able to be interpreted as an economic variable. In order to be able to describe QoS more succinctly and to be able to measure it, we will enquire closely into a particular economic theory, Lancaster's theory of consumer demand, with the aim of applying it to address the above mentioned concerns related to QoS.

Second, concepts from economics can be applied to solve various problems associated with the sharing of underlying computing and communication resources in an ODS. This economic-based approach to resource allocation mechanisms in computer systems is not new (e.g. [30], [47], [88] and [149]). However, previous studies have predominantly focused on resource allocation mechanisms in which knowledge about the state of the system can be regarded as perfect. In our view, this is not realistic in the context of ODSs. The novelty of our approach is in introducing the notion of uncertainty, which is inherent in ODSs, into a model of interactions between users and service providers in such systems. We will identify a particular class of services in ODSs for which uncertainty can be a significant problem, especially in terms of delivery of an agreed upon QoS. To this end we will endeavour to apply an economic agency theory to design contracts which will

govern service delivery in the presence of uncertainty in ODSs.

Concentration on the *business* related enterprise aspects of ODSs is motivated by the fact that an ODS architecture should provide additional commercial value to its owners and users, in terms of supporting the electronic inter-organisational business dealings. Since business contracts play an important role in such dealings, we will develop a business contract architecture that supports distributed processing across organisational boundaries. We will study the role of contracts from economic (i.e. agency theory and transaction cost economics viewpoints), legal and business perspectives and try to incorporate contracts into a supporting architecture for ODSs.

1.3 Thesis aim

The goal of this thesis is thus to provide a better understanding of how relevant concepts from the field of economics and business can be applied to the new, enterprise-related concerns of ODSs. In respect to this, part of the thesis is exploratory: we feel this is needed in order to address the lack of *any* work in this area at present (as is done in Chapter 3). In the course of this, we will also demonstrate how such economic and business concepts can be positioned within the architectural framework of the RM-ODP [64], [65], [66], i.e. how they can be used to enrich and populate the framework of the RM-ODP enterprise viewpoint.

In pursuing this goal we will develop the following.

1. Guidelines for how relevant economic theories can be applied to the specific enterprise concerns in ODSs. These guidelines can help enterprise strategists, managers and designers in selecting appropriate structures for distributed systems within their evolving organisations.
2. A general, service-independent framework for describing QoS and a generic methodology to measure it, based on the user perspective.
3. An analytical model which can be used to design optimal contracts for service delivery between users and providers in the presence of uncertainty. This can be particu-

larly suitable for a large class of services in ODS for which uncertainty represents a serious problem (e.g. reducing economic efficiency).

4. A business contract architecture which can be used to extend current ODS architectures so that business contracts can be supported. A prototype implementation of this architecture will also be presented.

1.4 The structure of the thesis.

In chapter 2 we highlight new characteristics of ODSs. It is shown that in addition to new technical challenges, ODSs embrace a number of non-technological (e.g. human, organisational) issues, which bring a variety of additional concerns. These are within the scope of non-computing disciplines such as business, economics, sociology, psychology and other social sciences, and need to be recognised in the specification of new systems and services. These concerns should be addressed through the use of appropriate concepts, principles and rules from these disciplines and utilised when specifying a distributed application or system.

In this chapter we also outline basic RM-ODP concepts, particularly those which fall into the so called enterprise viewpoint. The use of the enterprise concepts is illustrated with two examples: *i)* a Tourist Information Service, as a representative of an application domain and *ii)* the RM-ODP trader service, as a representative of the RM-ODP standardised service[1]. We then discuss the relationships between notions from the fields of economics, business, sociology and psychology, and enterprise issues of ODS architectures.

In chapter 3, the focus on enterprise aspects is restricted to the economic and business characteristics of ODSs. We outline those economic theories that we find relevant for ODSs and study how they can be applied to different economic related problems of ODSs. Additionally, some concepts from business practices (e.g. strategic management) are summarised as we argue that these also need to be considered when designing a commercially acceptable ODS architecture.

1. These two services are often used in this thesis, to illustrate many of the concepts and proposals discussed.

We begin this chapter with an outline of the basic concepts of traditional microeconomics, with the aim of emphasizing its inadequacies in addressing a large class of economic related characteristics of ODSs. We then illustrate how some other economic theories can be applied to address these inadequacies and are thus more appropriate for ODSs. For example, Lancaster's theory of consumer demand can address the notion of quality of service, and economic agency theory can be applied to incorporate uncertainty in designing contracts between parties in ODSs. In each of the sections of this chapter, which deal with the corresponding economic theories, we summarize how these specific economic theories can be applied to relevant problems within the domain of ODSs (some of these theories are analysed in more depth in chapters 4, 5 and 6). The chapter closes with a discussion on how ODSs themselves can influence changes in markets and organisations.

In chapter 4, we present a framework for describing and measuring QoS. We begin this chapter by first showing why traditional treatment of QoS in telecommunications is not adequate to address the requirements of ODSs. The emphasis should be more directed towards user-defined QoS specifications and performance measures. It is then shown how a service-independent QoS framework, which takes into account economics of quality and user-driven specification, can be developed with Lancaster's theory as a starting point. This framework can be used to identify user-defined QoS aspects and technology-related QoS characteristics.

However, in order to better reflect users' satisfaction with particular service aspects, i.e. to determine weights of certain service parameters, a paradigm from market research, conjoint analysis can be used. We highlight benefits and possible difficulties with this methodology and consider how a more general theory, information integration theory can be used to address the strong assumptions adopted by conjoint analysis.

In chapter 5 we outline the major sources of uncertainty of QoS delivery in ODSs and identify a specific class of services for which this problem can be emphasized. It is then shown how by designing appropriate contracts one can alleviate uncertainty. Such contracts should take into account different variables, e.g. players' attitudes towards risk, in-

centive payments and exogenous risk. Agency theory can be used to guide the design of such contracts. We demonstrate how this theory can be applied to the problem of QoS delivery in the context of ODSs and develop a formal model of a particular type of principal-agent scenario in the context of service provision in ODSs. We also discuss potential difficulties with this approach.

The theme of chapter 6 is the development of a business contract architecture for an ODS. This architecture aims at facilitating interactions between enterprises in electronic business dealings and thus alleviating uncertainty where it can potentially arise. This problem domain is complementary to the one from the previous chapter which focused on the economic role of contracts (as a mechanism that ensures optimal spread of benefits between interacting parties in an ODS). The idea being that, once a quantitative model for designing contracts which cover QoS delivery is developed, it becomes necessary that the concept of contract be supported within an ODS architecture.

We first develop a generic business contract framework, based on an economic treatment of contracts as well as relevant business practices and a supporting legal framework (business contract law). The purpose of this is to provide guidelines for an extension of the capabilities of current ODS architectures towards support for more coherent inter-organisational electronic business transactions. This framework is used to establish the core of the corresponding business contract architecture. We illustrate the applicability of our contract architecture with an example of an electronic stock exchange and the RM-ODP trader component.

Furthermore, this architecture is then used as a starting point to introduce the architectural notion of QoS and thus to position QoS within the specification and infrastructure models of a specific ODS architecture. We demonstrate how a business contract can be transformed into the specification part of this model and how, via QoS specification, this can be mapped onto the underlying resource aspects.

The major concepts of the business contract architecture developed in this chapter can be used as a starting point to design and develop the corresponding implementation of the

architecture. In chapter 7 we present a prototype implementation, developed by using a specific distributed programming environment, conformant with the OMG CORBA specification [62].

In chapter 8 we conclude with a discussion of the main contributions of this dissertation and outline some possible directions for future research.

CHAPTER 2

Open Distributed Systems (ODSs): Enterprise Issues

In this chapter we highlight new characteristics of emerging information systems, *open distributed systems* (ODSs), and the challenges that these face as a result of the need to appropriately address growing expectations and requirements of the contemporary information society. We emphasize the fact that an important class of such challenges results from a number of non-technological issues, which bring a variety of new concerns. These are frequently placed under the umbrella of *enterprise* issues.

After a brief introduction to the evolution of information technology (IT) towards ODSs (in section 2.1), we outline indicators of current changes in the IT industry (section 2.2). The third section analyses ODSs in terms of an interplay between technological and market issues. This is followed by the description of international standardisation on Open Distributed Processing (ODP) in section 2.4, and the ODP enterprise viewpoint, which indeed represents the problem domain of this thesis, in section 2.5. We then discuss the relationships between concepts and results from the fields of economics and business as well as sociology and psychology, and enterprise issues of ODS architectures, in sections 2.6 and 2.7 respectively.

2.1 Introduction

Through the evolutionary process, human society has undergone various developmental stages, each of which has been characterised by a particular form of production, accompanied by specific tools, skills and knowledge (technology). These factors have been a determinant of man as an individual and social being. While there are different theories of what can be regarded as the engine of this evolution[1], it can be said that the specialisation of labour has contributed to the economic growth of human society. This specialisation has led to what Adam Smith referred to as the increased 'wealth of nations' [135]. In its historical development, by increasing the level of specialisation, society has progressed from a tribal society, through an agricultural and industrial stage to the present age of information [155]. Each of these developmental stages can be characterised with specific characteristics of individuals (e.g. values, skills) and their relationships within different social groups (e.g. families, organisations).

Information technology has brought about significant changes to both individual and social aspects of people's lives. The early stages of the information age can be attributed to the technological advances in electronics, such as vacuum tubes in the fifties and transistors in the sixties. This was followed by the emergence of integrated circuits and the first microprocessors in the seventies. The eighties can be characterised as an era of dramatic growth in computer communications networks, both Local Area Networks (LAN) and Wide Area Networks (WAN)[142]. This was the result of further technological advances in computing and telecommunications which enabled more flexible utilisation of computing resources located at several points. With this, came the first new applications such as remote log-in, the e-mail facility, remote database access and electronic data exchange (EDI). This trend continued into the late eighties and was augmented by the emergence of the first networked distributed systems. Distributed systems are essentially an extension of computer networks, characterised by cooperation of processes in order to achieve a common goal [74]. We have witnessed distributed system products such as Amoeba,

1. For example, the Marxian conception is based on the struggle between contending classes, while Adam Smith's theory of historical evolution centres around human nature driven by the desire for self-betterment, guided by the faculties of reason, as the primal moving agency [44].

MACH, Athena and others [143]. These developments signified a new era in commercial computing - the era of *distributed processing*. According to [66], distributed processing comprises that class of information processing activities in which discrete components may be located at more than one location or where there is any reason which necessitates explicit communication among components. Distributed systems have revealed to users new possibilities for further advancing the wealth, especially in terms of increasing efficiency for existing business procedures and providing new ways of doing business.

However, the present time brings even more exciting technological advances in both computing and telecommunications. These are juxtaposed with the increasing role of market forces which lead towards the decreased cost of IT products. Coupled with new regulatory policies, these factors can explain a shift in the direction of the IT industry and telecommunications - a trend towards *open* distributed systems (ODSs). Our view of openness encompasses both technical aspects (e.g. interworking of different technologies) and market aspects (e.g. open access to information services publicly offered and interworking across organisational boundaries), as will be elaborated in sections 2.2 and 2.3. It is expected that a trend towards ODSs will dominate the late nineties and beyond. We are witnessing the beginning of this era, through concepts such as tele-commuting, electronic commerce, distance education and mobile computing. Hence it is evident that ODSs will further change the way we live both at an individual and social level. Some of the new, technology influenced aspects are discussed in more detail in sections 2.6-2.7.

2.2 Indicators of current changes in IT industry trends

There are several indicators which characterise the shift in IT industry towards openness.

Firstly, owing to lower per-unit price of CPU speed, IT systems are now becoming more accessible to the *community at large*. This is evidenced by the fact that they are now more widely employed, infiltrating the domains of residential customers and small businesses as well as their traditional deployment in medium-size businesses and large, global corporations. Moreover, the capabilities of these new systems far out perform older (less effective and more expensive) systems, allowing them to be used in very complex

applications. A decade ago, such applications were the privilege of large organisations only. Examples would be the increased use of powerful desk top publishing tools (including multimedia) and implementation of local networks in small offices and organisations. Hence, there is rapid growth of installed computer resources across a very large (even global) customer base. This, along with a growing demand for more sharing and exchanging of services and information, will decentralise processing of information and execution of services.

Secondly, armed with an awareness of the importance of IT in the mission of the organisation, users are now increasingly requiring *interworking* of their existing heterogeneous systems so that a seamless, enterprise-wide information infrastructure can be delivered. Rather than viewing the IT system as a collection of islands of different systems and technologies, IT systems ought to appear as a whole, forming an integral part of enterprises. Our view of interworking encompasses interworking between different technologies but also *portability* between different vendor products. For example, installation of the same package on a different computer platform should make the differences between underlying operating systems, computer architecture and communication protocols transparent to the user.

Thirdly, the proliferation of networking has revealed to the user the possibilities of utilising services and products globally available, in a more flexible and cost effective way. Hence, users are demanding *open access* to information service markets where they can select information or services as they require and according to their Quality of Service (QoS) and price requirements. In this way, they can increasingly utilise services developed by others. This indeed is in the spirit of labour specialisation that Adam Smith identified as the major factor for the wealth of nations.

Fourth, users are becoming the central factor in determining the meaning of the notion, QoS. They are now in a position to dictate what they require from services and what quality aspects are relevant for them. QoS becomes an *end-user* issue determined on an application to application basis. This means that ODSs demand the need for a broader scope of QoS issues which include both *technical* and *economic* factors. Economic factors arise

from the fact that users' enterprise objectives often depend on QoS levels. For consumers of service, QoS determines satisfaction, and for service producers it is an essential component of their competitiveness.

Finally, bearing in mind the serious flaws of IT solutions which have been adopted previously, users are now more likely to choose solutions which place an emphasis on *evolvability* of their systems. In other words, new systems should interoperate with legacy systems but should also be capable of gracefully evolving as new technologies or products emerge.

Therefore, in a highly competitive environment of today, users have the dominant role: they demand an access to, and efficient utilisation of resources, information and services regardless of the underlying computer platform, communication protocol, software product or administrative constraints. In order for the users' business requirements to be met, and, owing to available technologies, information systems evolve towards *open distributed systems* - a point of convergence between previously two separate technologies, computing and telecommunications.

2.3 Open distributed systems

As opposed to early distributed systems, which could be regarded as closed since they have been implemented within the administrative boundaries of different organisations and based on proprietary solutions, emerging distributed systems are becoming more *open*, allowing increased interworking, portability and open access to the community at large. This is made possible owing to the rapid development of technologies which provide an increasing processing speed (RISC architectures, 64 bit machines), faster transmission media (fibre optic) and broadband switching and transmission, as well as more advanced and sophisticated services. These developments are juxtaposed with the increasing role of market forces which lead towards the decreased cost of IT products. Coupled with new regulatory policies which promote competition (especially in telecommunications) [112], these factors can explain a shift in the direction of the IT industry and telecommunications. These two industries are increasingly merging, and ODSs

play a significant role in this (Fig. 2.1). The synergy of technological advances and new regulatory policies contribute to the positive changes which have direct impact upon service consumers. The ultimate goals of these changes are a wider choice of products and services, better quality/performance at the lower price as well as an increased availability of these technologies to the community at large. For example, we are witnessing the fact that revolutionary changes in telecommunications, such as Intelligent Network (IN) services, broadband technologies and sophisticated customer premises equipment result in increasing customer expectations and awareness about competitiveness of new telecommunications and computer (e.g. information) markets.

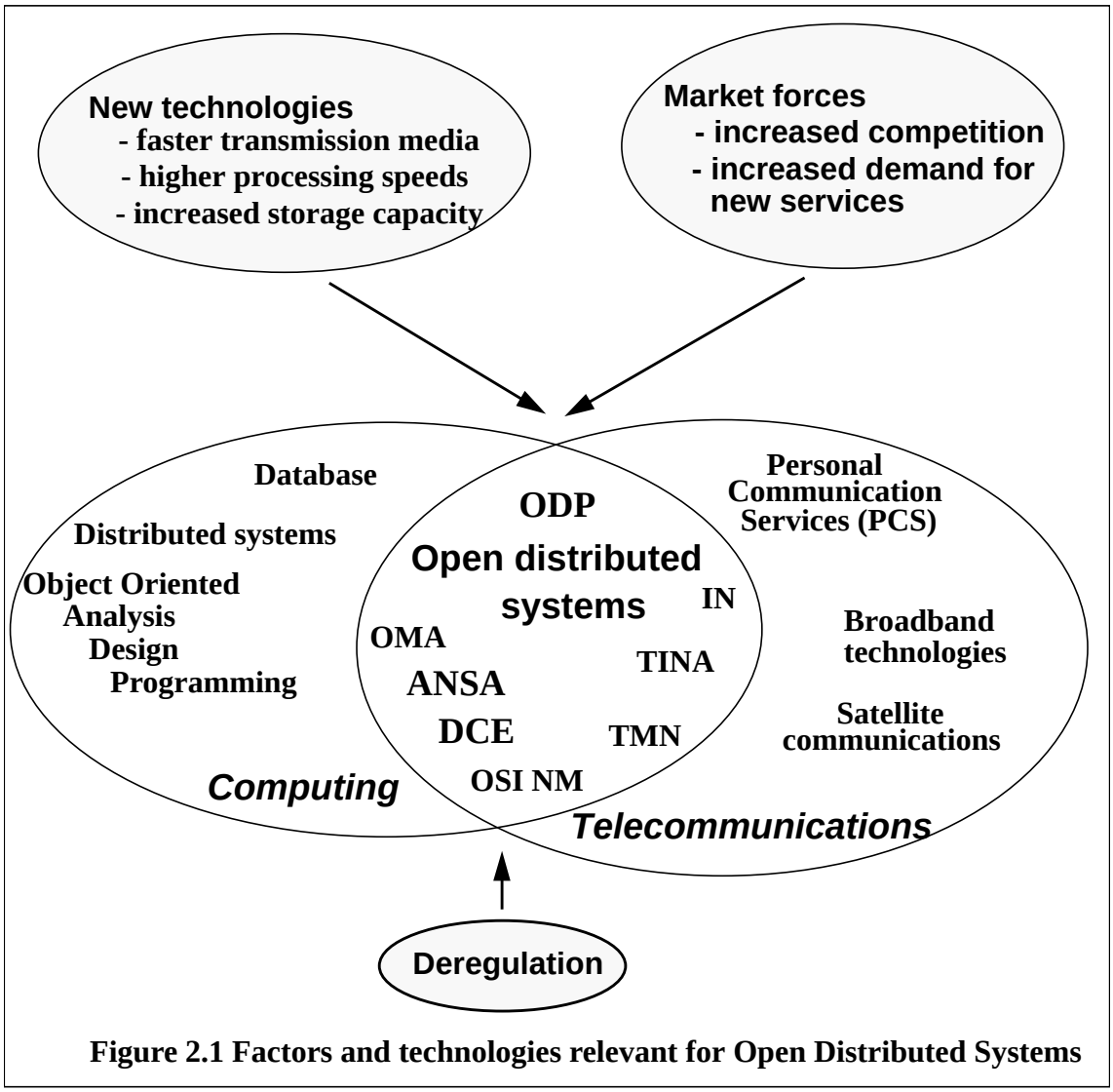

Figure 2.1 Factors and technologies relevant for Open Distributed Systems

A number of commercial efforts in the communications and computer industries prove the awareness of service providers of the importance of these radical changes in both indus-

tries. Their common denominator is the provision of distributed environments which are more open: they provide an increased *interworking* between heterogeneous products/ technologies, interoperation of different *administrative* domains, and more *open access* to the public. Examples of ODSs from the computer world are architectural models (and associated supporting packages) such as ANSA [3], DCE [126], OMG's OMA [108] and OSI-Management Framework [79]. In the telecommunications arena the best examples of related ODSs are the maturing IN concept [27], TINA[2] initiative [9] and TMN [31].

In addition to the growing number of commercial products along these lines, an important contribution also comes from international standard organisations. The joint work of International Organisation for Standardisation (ISO) and the International Telecommunication Union (ITU-T) on Open Distributed Processing (ODP) [64], [65], [66], addresses most of the issues mentioned above. Its aim is to develop a unifying framework which will provide long term interworking between various technologies, different vendor products and existing and future technologies. ODP can be seen as an important catalyst in gathering momentum for a more increased development of information markets. These will in turn bring a plethora of new economic issues of concern for distributed system and application planers, developers and implementers.

Therefore, ODSs can provide a basis for new cooperative environments which can benefit all parties utilising the capabilities of the emerging systems. To support these new features, ODSs offer a much wider range of mechanisms than traditional client-server solutions. For example, applications such as information searching, Computer Supported Cooperative Work (CSCW), multimedia conference and nomadic communications require additional, more sophisticated mechanisms.

There are several *technical characteristics* which need to be realised within ODSs so that the objectives of IT suppliers can be achieved in response to increased user pressures. This includes:

2. Telecommunication Information Networking Architecture is an international telecommunication initiative, which aims to provide an architecture that would provide information at any place at any time in any form and in any volume [9].

- support for *distribution*, i.e. hiding the complexity of technical details associated with the distribution such as location of service or information, access mechanisms for different computer platforms as well as failure and recovery of hardware and software components
- *integration* between different computer system architectures and resources, possibly with different performance characteristics
- *portability* of the application across heterogeneous platforms
- support for *interworking* within and across organisational boundaries; this requires appropriate mechanisms to support coordination of activities of the parties involved
- *s*upport for many different *types* of services and applications, e.g telecommunications, office, factory and general data processing
- *scalability* of systems so that the information, services and resources can be accessible at any remote location as easily as if they resided locally
- capability to include installed IT base (*legacy* systems) and thus keep present investments
- open *access* to information services markets, resulting in more choice to users of different QoS levels offered
- capability to respond to more stringent and demanding *QoS requirements*, whereby, QoS should be considered on an *end-to-end* basis, as perceived by the application, (traditional communication QoS is just one element of this)
- flexibility in terms of capability for *QoS negotiation, QoS guarantees* and *dynamic QoS* aspects
- provision of all the above while preserving security and privacy
- multitude of management requirements associated with each of the above characteristics.

These technical features require a certain level of consensus among IT suppliers, i.e. an agreement on common structuring across IT suppliers as well as understanding and exploitation of common structures among users [61]. ODP can provide such consensus.

2.4 Open Distributed Processing (ODP)

As noted previously, many organisations and individual customers have recognised the benefits of distributed systems and have started to employ them. However, there is still an obstacle which needs to be overcome in order to provide a more open solution (according to the description given in the previous section) and thus respond to important customer expectations: a variety of different products and solutions which need to be aligned. As a prerequisite for overcoming this obstacle, it is desirable that distributed systems possess a number of common technical properties, as outlined above. This requires some level of agreement on common structuring among IT vendors and an understanding and exploitation of that common structure among users [61]. The Open Distributed Processing (ODP) standard can facilitate this. It describes systems that support heterogeneous distributed processing both within and between organisations through the use of a common interaction model. Its objective is to provide a means to establish necessary common structuring as well as to understand and exploit it. The Reference Model for ODP provides such a framework: an architecture within which support for distribution, interworking, interoperability and portability can be integrated.

2.4.1 The Reference Model for Open Distributed Processing (RM-ODP)

The Reference Model for ODP (RM-ODP) provides a framework for the standardisation of Open Distributed Processing by creating an architecture which supports distribution, interworking, interoperability and portability [118].

The RM-ODP provides a common language for describing distributed systems as well as a description of common structures, using the language. It defines the technical basis for ODP standards, and relates this to other ISO Reference Models and existing standards, including data communications and management standards. The RM-ODP has established a style and basis for system design and implementation, allowing use of common components and common designs, methods and representations [61]. It aims to provide consistency among multiple standards developed separately. It also sets the technical agenda for

future, more detailed standardisation [82]; standards for specific purposes can be produced to cover a specialised field of application, such as TINA for telecommunications.

It is important to mention that several current commercial initiatives and products are closely associated with the ODP recommendations, e.g TINA, ANSAware[3] and OMG's CORBA[4] conformant systems.

2.4.2 The RM-ODP Architectural Framework

Two main structuring approaches used in the ODP architecture are the definition of *viewpoints* and the definition of *distribution transparencies*.

To master the complexity of an ODP system (and applications which run on such a system), RM-ODP adopts the so-called viewpoint approach. According to this, the system can be considered from different viewpoints, where each viewpoint represents a different abstraction of the system (or an application), that addresses a separate concern of a relevant party(ies) involved. Five such viewpoints are identified, viz enterprise, information, computational, engineering and technology viewpoints, as described in [64], [65], [66].

The *enterprise* viewpoint focuses on the expression of *purpose*, *policy* and *boundary* (i.e. limits to openness) for an ODS. It also recognises that there may be conflicts of purpose or policy at a boundary which have to be modelled [61]. Hence, the enterprise viewpoint addresses strategic and policy issues, for example the role which an information system or service has within an enterprise and how it relates to enterprise objectives. These basic issues represent a starting point for defining corresponding requirements and policy statements. The enterprise specification of service should identify the agents, their roles and performative actions in provision of a service and related policy statements. Presently, the enterprise specification is expressed in natural language. This viewpoint is within the scope of concern for enterprise strategists and enterprise requirement definers.

3. Infrastructure which implements principles from Advanced Networked Systems Architecture (ANSA) [3].
4. Object Management Group's (OMG) Common Object Request Broker Architecture [109].

The *information* viewpoint is concerned with the expression of information and information processing functions. Information specification deals with information objects which model real world entities (e.g. agents, artifacts and others from the enterprise specification) and transformations that occur to these objects (which are in fact, the actions defined in the enterprise specification). This viewpoint is of interest to the information modeller.

The *computational* viewpoint addresses the functional decomposition of an ODP system, and, interworking and portability issues [61]. The computational specification has to provide service functionality and information object definitions according to the description in the information specification. In the computational language, service functionality is realised by the operations of computational interfaces, and information objects are represented using the basic data types defined in the computational language (e.g. ANSAware, DCE or CORBA Interface Definition Language). This viewpoint is of concern for the application programmers.

The engineering viewpoint focuses on the expression of the infrastructure required to support distributed processing in an open environment; this is of concern for system engineers.

The technology viewpoint is related to the expression of underlying technologies that support the system.

The relationships between these viewpoints and traditional software engineering process are depicted in Fig. 2.2 [117].

It is worth noting that the different viewpoints are independent views of the same system but are related to each other [20]. The ODP enterprise viewpoint serves as the basis for specifying enterprise goals (and associated policy statements) on which all other viewpoint specifications will directly or indirectly depend.

The ODP viewpoints are used to derive different models of the same system. This is facilitated through the use of the corresponding viewpoint languages. These consist of the

specific set of concepts, structures and rules which can be used to specify each of the models.

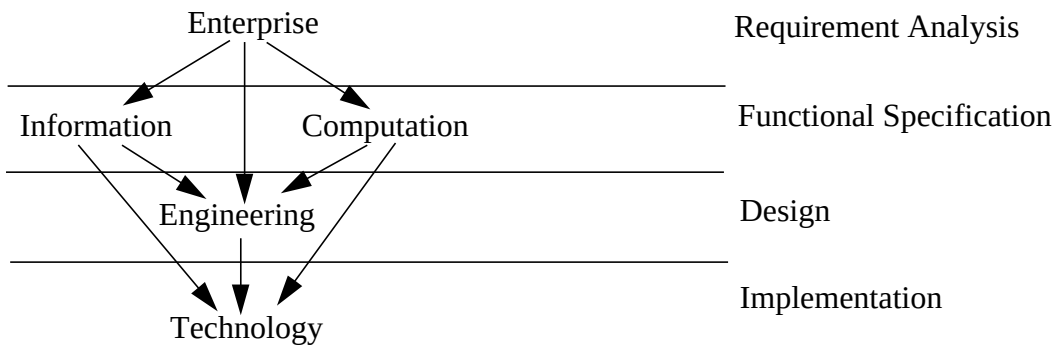

Figure 2.2 RM-ODP Viewpoints and traditional Software Engineering process

Another structuring concept adopted within RM-ODP is the *distribution transparency*. This concept can be exploited to hide the consequences of distribution from both the applications programmer and the user. The introduction of distributed transparencies was born out of the need to address a number of problems in distributed systems which arise from the distribution. For example, in a distributed system, the systems components are heterogeneous, they can fail independently, they are at different and possibly varying locations. These problems can either be solved as a part of the application design, or standard solutions can be selected, based on best practice [82]. In the latter case, the application programmer can be equipped with a standard mechanism which can provide a transparency, so that the particular distribution problem becomes transparent. Thus, the term distribution transparency. It is worth noting that distribution transparency also promotes software reuse.

The transparencies defined in RM-ODP are access, failure, location, migration, relocation, replication, persistence and transaction transparency. A detailed description of these transparencies can be found in [66].

2.4.3 The RM-ODP Trader component: basic concepts

One of the central services in an ODS which supports establishment of the market of in-

formation services is a *trading* service. This service provides functionality similar to the concept of the familiar yellow pages directory issued by telecom providers. The trading service is implemented by the ODP component called Trader (a similar concept is also present in architectures such as ANSA and TINA, and is in the process of adoption within the OMG group). This component stores service offers exported by servers (or service providers, SPs), and provides a search capability to clients (or service requesters, SRs), so that advertised SPs' offers which match SRs' requests can be found and then imported (Fig. 2.3). Optionally, a trader can select from the set of possible matches the best offer for a SR, based on dynamic properties of SP offers (according to a predefined selection algorithm). Furthermore, traders can be linked together to interwork (form a federation) [11], in order to provide a wider market for exporters and a greater choice for importers.

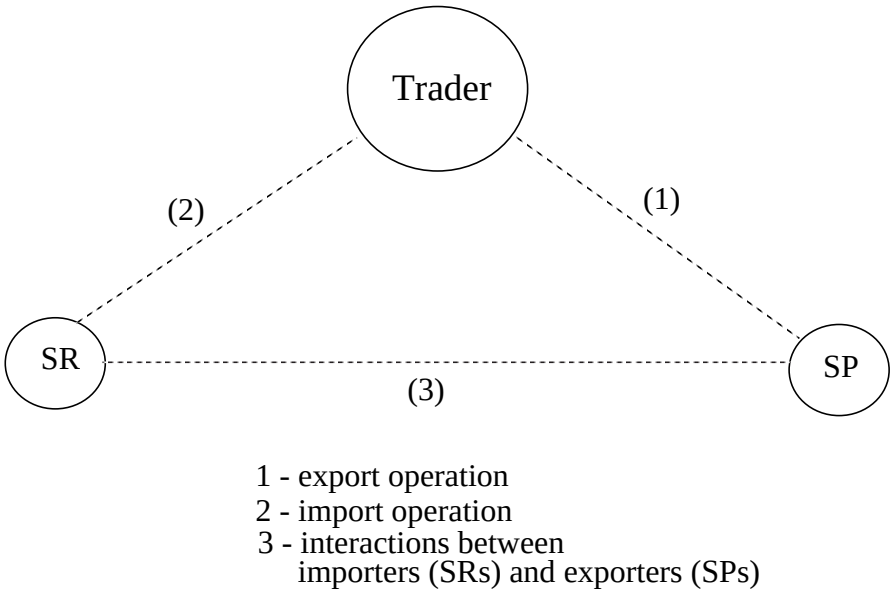

Figure 2.3 Interactions between the trader component and its users

The trader is one of the most important ODP functions, because it allows dynamic configuration and evolution of distributed systems. Because of its importance it is one of the first of ODP functions to be standardised [82].

In this thesis we shall often use the trader to illustrate many of the concepts and proposals discussed.

2.5 The RM-ODP Enterprise viewpoint scope

It is now recognised that information systems which would penetrate almost every aspect of our daily lives cannot be developed without appropriately recognising relevant non-technical issues. In general, the application designer should understand and require that an architecture does not rule out 'the proper balance between human, organisational and technological issues' [61]. If this is not accepted, we will increasingly face situations in which the technology (information systems) does not optimally match non-technical (common-life) realities. These non-technical issues are within the scope of disciplines such as economics, business, organisational theory, law and other social sciences. Hence, a need arises for these issues to be accounted for and to be explicitly addressed in the specification of an ODS or applications within it. A closer look at these areas is therefore a prerequisite in understanding how well an ODP conformant system addresses the environment in which such issues are relevant. To this end, it is necessary to identify, express and systematise a number of concepts important to all actors involved in the ODP system, designers, users, system owners, and place them in a proper architectural framework [61]. Examples of these concepts are, organisational structures, ownership, responsibility, obligation, information structures and control structures. This calls for a multi-disciplinary approach.

2.5.1 The RM-ODP enterprise language

The RM-ODP sheds light on a number of non-technical issues by placing them in a separate domain, the enterprise viewpoint. The ODP *enterprise language* identifies a limited set of concepts, principles and rules within the enterprise domain; those that are applicable to any problem domain. The aim of this set is to represent a basic framework from which social and organisational policies can be defined. In other words, the enterprise specification can be used to express the objectives and policy constraints on the system of interest [82].

The basic enterprise language concepts and structuring rules are as follows [66], [82].

- Performative action - an action which changes obligations, prohibitions and permission of objects.
- Agents - ‘active’ objects which initiate performative actions (i.e. actions that change policy, such as creating an obligation or revoking permission).
- Artifacts - ‘passive’ objects which do not initiate actions.
- Communities - group of agents formed for reasons of purpose.
- Roles of the agents, artifacts and communities expressed in terms of policies.
- Administrator - an agent which enforces policy statements with respect to a set of resources.
- Resource - an object whose existence or activity is subject to accounting by an administrator.
- Contract, which expresses the common goals and responsibilities which distinguish roles in a community, such as business and its customers.
- Federation - a particular kind of community. This is a coming-together of a number of groups answering to different authorities in order that they may cooperate to achieve some objective.
- Ownership of resources and responsibility for payment of goods and services in order to identify, for example, constraints on accounting and security mechanisms within the underlying infrastructure.

These basic enterprise concepts can be extended with specific notions from other application areas when relevant guidelines, recommendations or standards need to be produced. A particular area of interest of this thesis is *economics* and its relevance to the ODP enterprise viewpoint. Other disciplines will also be mentioned briefly in this chapter (section 2.7).

2.5.2 Examples of the use of the RM-ODP enterprise language

In order to illustrate the use of the ODP enterprise language specification we choose one

specific application (envisaged to be widely available in the near future), a tourist information service (TIS) [98] and one ODP component, the ODP Trader.

Example of an application domain: Tourist Information Service

We assume the following scenario. A TIS allows users to have access from their homes, or their organisations, to a multimedia tourist information directory. This directory has a wide range of related information, a subset of which is presently available from traditional travel agents (an example is shown in Fig. 2.4). Users will normally be able to perform the following functions:

- searching through directory information in order to find particular types of information
- information presentation of particular multimedia information found after searching, e.g. a video presentation of a particular resort, hotel rooms, surrounding areas, potential entertainment guide, accompanied with high-quality audio information, music etc.
- booking of specific accommodation, flights etc.
- access to various forms of billing information
- customising of the TIS according to customers' own needs.

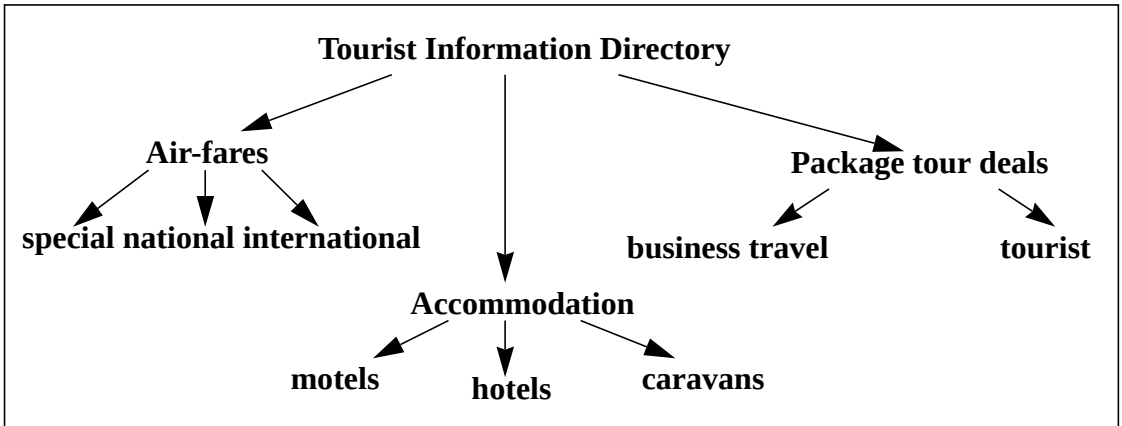

Figure 2.4 Information available from electronic Tourist Information Service

Following the enterprise viewpoint, major *agents*[5] centred around this service and their relationships can be identified (one possible scenario is depicted in Fig. 2.5):

- An end-user of the TIS, who is directly using it and who pays for its usage.
- The TIS provider - an agent who owns and offers the TIS. Examples are: a large (possibly global) organisation, either being a private company (such as a tourist agent, air line etc.) or public authority (such as a government tourist organisation). The TIS provider may or may not be an *owner* of network *resources*. In the latter case, it would normally purchase network resources or services from a carrier and uses them to offer its own TIS, for which its *charging policies* apply.
- A network provider - an agent who provides broadband transport services, necessary for the implementation of multimedia services. The network provider is the *owner* of network resources and services whose use is the subject of its *charging policies*.
- A TIS subscriber (a subcontractor of the TIS provider) - an agent who provides primary tourist services, such as air-fares, accommodation, car-rentals etc.

Each agent has a set of enterprise objectives, subject to constraints and policies set by national or international regulatory authorities. Possible enterprise objectives can be the maximisation of variables such as profit, profit rate, enterprise growth opportunities, QoS value, performance/cost ratio and an increase in enterprise prestige [102].

Being an objective of the *community*, service provision should be seen as governed by a set of *policies*. An important set of *policy statements* which are addressed in this thesis relate to enterprise specific QoS issues, regarded as an important business link between agents.

Interactions between agents are determined by two types of *performative actions*: one agent provides a service to another, for which the latter pays a certain amount of money. Therefore, one agent is a service ‘producer’, while the other is a service ‘consumer’. An agent can have both consumer and producer *roles*; for example a service provider normally has a consumer role with respect to a carrier, and a producer role with respect to end-

5. ODP enterprise language concepts are given in italics.

users or service subscribers. Basically, a consumer of a service is concerned with the value obtained from this service (which directly depends on the level of QoS provided), and a service producer, with the (monetary) value received from the consumer. Thus, in the service provision process, this <QoS,price> pair binds agents which take part in this process (as depicted in Fig. 2.5). The practical implementation of this binding process is a *policy statement i*n terms of a *contract* formed between agents after contract negotiations are carried out. This contract covers issues such as guaranteed levels of QoS, corresponding price and responsibilities of agents, as described in more details in chapters 5 and 6.

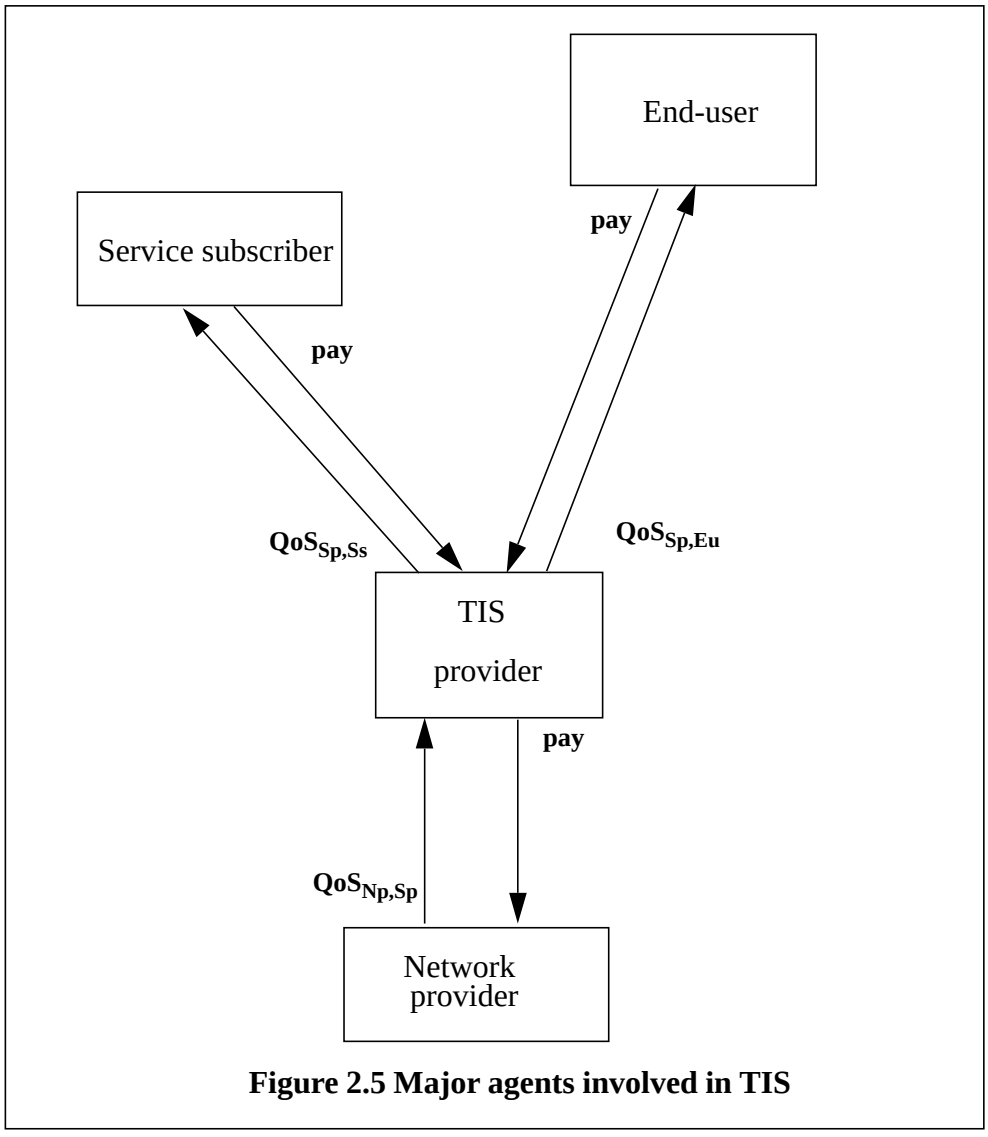

Figure 2.5 Major agents involved in TIS

Example of an ODS service: the RM-ODP Trader component

From the enterprise viewpoint, the *owner* of trader identifies the *policies* to provide guidelines to achieve the objectives of the functional requirements of trading. The objectives are [11]:

- to provide service information to clients
- to enable clients to be configured without prior knowledge of required servers
- to provide the capability of interworking with other traders.

The trader policies encompass:

- general trader policies (e.g. for arbitration, security and remuneration)
- export action trader policies (e.g. service offer acceptance policy, service placement policy, storage policy)
- import action trader policies (e.g. local search policy, resource consumption policy).

Following the ODP enterprise language concepts, a trader is the centre of *community* which:

- is established for the *purpose* of trading
- consists of members with roles, e.g. *administrator* (enforces the polices), trader (stores and matches offers according to the policy), exporters and importers
- is governed by a trading *policy.*

Members of trading community are obliged to obey the *policy* rules.

Several traders can interwork (be federated). A group of interworking traders:

- forms a *community* of traders with their respective importers and exporters
- one trader is not obliged to perform an activity of another trader
- each trader must have complete control of its own trader policies

- when a trader joins a group of interworking traders, all members of the trader's community are obliged to be part of the group.

In the case of federation, each trader has its own policy with regard to:

- the propagation rules of a search request through the linked traders
- the propagation rules to be considered in exporting service offer to other linked traders.

A trader's interworking requirements are provided by:

- global search policy - rules to guide the use of links of a a given trader for an import operation
- domain boundary crossing policy - rules to guide the trader in crossing boundaries of domains (e.g. type domains, security domains)
- resource consumption policy - rules to guide the use of resources of a given trader (e.g. the maximum search time for an import operation, the maximum cost of search and the maximum number of traders to be visited).

2.5.3 Enterprise viewpoint modelling

As discussed previously, the major focus of the enterprise viewpoint centres on purposeful interaction. This includes interaction between humans, between humans and computer (or communication) resources and between these resources themselves. Various approaches can be taken to describing and modelling these interactions in a more general fashion. An approach adopted in [61] is to view an interaction as a sequence of exchanges where information flows for some purpose. Such exchanges are termed conversations and are analysed in depth by using the concepts of linguistic philosophy and linguistics. Conversations are used in the context of the enterprise viewpoint to elicit:

1. the domain of interest
2. the entities and their relationships within the domain
3. non-operational requirements

4. goals.

This approach is general enough to be applied as an enterprise modelling methodology to any specific domain of interest for ODSs. In the following section, we will use this approach as a starting point to describe *economic* and *business domain*s which are the focus of this thesis.

2.6 Economics and business domains of the enterprise viewpoint

As a prerequisite for a sound enterprise modelling, the information strategists and enterprise requirement definers need to be armed with a sufficiently thorough understanding of the relevant fields of economics and business. This will provide them with the following capabilities.

- Modelling of a *structure* of the enterprise to the extent which is relevant for the information system implemented (or to be implemented). This is an important task for identifying the role of IT as a competitive element for the enterprise.
- Understanding of the role of IT as a cause and consequence of change in the structure of organisations.
- Relating *business* requirements to the technical design of the system, to provide guidelines for the design of a system or a service which best suits the environment in which it would be used. The intensive work on different business process modelling methodologies augments the importance of this issue [12], [110], [128], [132], [148].
- Modelling *interactions* between agents in an ODS, for example, deriving optimal contracts between actors in the presence of conflicting objectives and uncertainty.
- Predicting moves and actions of potential rivals so that an appropriate *strategy* can be developed, taking these into account.

Once these enterprise related issues are addressed, it is possible to embark on the process of the design of an ODS system or application.

According to the previous four-step approach, the interactions among economic actors within an ODS environment can be described with the following steps.

2.6.1 The domain of interest

We have shown how ODSs, with underlying computing and communication technologies, will be a major enabler for new information environments. It is expected that this can happen within organisations and across markets. As a result, there will be increased interaction between a multitude of different parties at various levels. Our focus is on the *economic* interactions which arise in ODSs, as a result of the fact that users (i.e. economic actors) utilise shared system resources in the course of pursuing their (economic-driven) objectives. A common denominator for these interactions should be seeking mechanisms which provide efficient allocation of underlying resources for the benefit of all; a scenario which is the subject of study in economics. Hence, our *domain of interest* covers economic (and business) type of interactions in ODSs.

The fundamental, economic-specific characteristics of ODSs (as identified in section 2.2) are as follows.

- An *open access* for information services to the community at large. This will result in the use of resources by different types of users (e.g. residential and business) with different objectives and different characteristics (e.g. preferences and endowments).
- Simultaneous existence of information services of *highly differentiated types* and different *levels of quality.*
- *A cooperative* and *competitive* environment in which service provision can be done by different service providers individually, or in cooperation with others.
- Emergence of a *global* market for information services. A component which can support this is the *ODP trader*, and similar commercial concepts.
- *Users' business needs* are the driving force for suppliers: when there is a demand for a service, the suppliers increasingly compete with their QoS offerings. An important new factor is the economics of quality.

Therefore, these characteristics imply an increased interaction between different players while utilising the shared resources of an ODS. It is more likely that there will be a scarcity rather than an abundance of the resources of a system. For example, despite an increasing bandwidth capability of the Internet, many new services enabled by the capabilities of World Wide Web facility [14], [72], have dramatically increased traffic on the Internet and are still demanding a higher bandwidth. In order to provide effective and efficient utilisation of Internet resources, the appropriate pricing mechanisms are required, as a resource allocation mechanism [8], [86]. Similarly, with a wider use of ODSs, a number of new economic related problems will arise as a result [100]. This sets our motivation for seeking solutions to various ODS resource problems in relevant economic theories.

2.6.2 The entities and their relationships within the domain

A principal type of economic interaction is a service provision, in which a minimum of two *entities* are involved, a service consumer and service producer. Their *relationships* can be described by the exchange that takes place (i.e. economic transaction), through which the producer transfers rights to the consumer to use his services in return for some payment. This transaction can be regarded as an atomic building block of a multitude of more complex interactions and relationships which arise in information markets. It is important to note that business contracts are designed to formally (and legally) specify such relationships and the roles of parties in economic transactions, and that their structure is normally domain dependent.

2.6.3 Non-operational requirements

Two important non-operational requirements of concern for economic interactions in the ODS context are Quality of Service and price. These are typically contract terms which directly specify transactions between related parties. Additionally, one should also consider other variables, which do not explicitly appear in the contract but represent econom-

ic constraints on the parties participating in interactions, e.g. different aspects of competitors, such as their number and strength, the parties' technology and capital available.

2.6.4 Goals

Each party, whether it be an individual or organisation has one or more *objectives*. It is normally assumed that each economic actor has the objective of profit maximisation. While this is the prevailing objective it may not be the sole objective, as will be discussed in subsection 3.6.3.

Economic specific issues of the ODSs will be analysed in more depth in the forthcoming chapters. We will now illustrate some other domains of interest, and their relation to the enterprise issues of ODSs.

2.7 Other enterprise domains

From the previous discussion, it can be concluded that new technological advances will enable penetration of computing and communications in many aspects of every day life (and will change these as well). Hence, in order to appropriately respond to the IT developments, one needs to understand and examine the interrelationships between technological trends and human factors [33], [49], [155]. In order to view these interrelationships in a holistic manner they should be considered within two contexts, the individual and the social context (as discussed in section 2.1). Accordingly, two broad categories of sciences can be distinguished i.e. the sciences which study man as an individual and the social sciences. The latter delve into the different interactions between people and their environment. These sciences are listed in Fig. 2.6, which also shows relationships between the two contexts (the sciences given in bold font style are dealt with in this thesis).

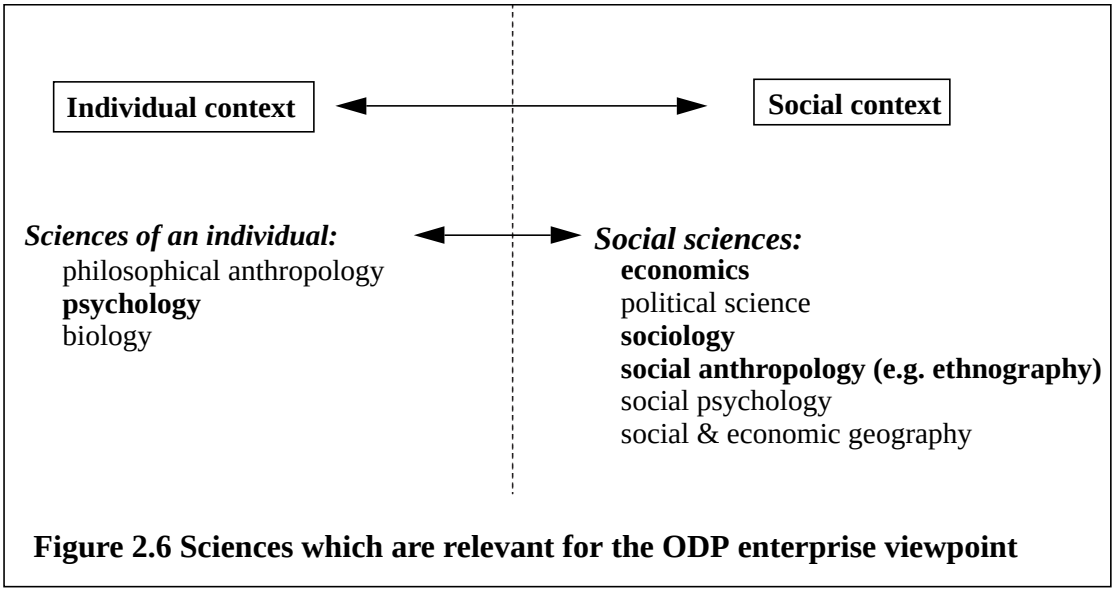

Figure 2.6 Sciences which are relevant for the ODP enterprise viewpoint

We have briefly mentioned some relation between ODSs and economics (as a special type of social science). These will be addressed in more details in the forthcoming chapters. In the following, we will be investigating contributions from some other social sciences such as sociology, social anthropology, and psychology.

2.7.1 Sociology and social anthropology

The contributions of sociology and anthropology to ODSs can be viewed in the description and analysis of the real world settings in which systems will be implemented [18]. While psychology has been helpful in deriving individualised, cognitive models of the user and his/her behaviour, many real life activities which involve interactions between people within groups require additional insights. A typical example would be work that is carried out in the social context, as a cooperative activity. The fact that work is a social activity should be kept in mind when designing supporting IT systems. These systems are usually termed Computer Supported Cooperative Work (CSCW). However, as found in the study reported in [18][6], the cooperative nature of work has been largely ignored in the current state of the art computer technology. For instance, existing word-processors provide excellent environments to help an author produce highly professional documents,

6. This study investigated the potential impact of CSCW on ODP.

and yet fail to consider the fact that most document production is highly collaborative [18]. To this end, the CSCW community has recognised that a multi-disciplinary approach should be taken, whereby contributions to the field are made by sciences, such as psychology, sociology, economics and organisational study.

In order to assist the development of CSCW systems so that they can correctly address the nature of work as being a social activity, the field of ethnography (which has its origins in social anthropology) holds a promise as a particularly suitable technique. An aspect of ethnography of special interest for IT professionals is its study of how work is organised as a social activity. The ethnographers observe workers in their environment in order to gain an understanding of the actual as opposed to formal work practices. For example, within the social context of work, there are many subtle interactions involving gestures, informal interchanges, but also implicit knowledge, that may be important factors for work. Hence, for applications such as CSCW, it is important to understand that the process of allocating tasks among individuals can be very flexible, often based on factors such as the current context and level of activity and not necessarily on prescribed roles and procedures. An advantage of ethnography is that it captures details which traditional requirement analysis may often miss. Ethnographic studies are helpful in informing the systems design process and may produce insights which contradict conventional thinking in systems design.

These and other more specific conclusions have been drawn from the work carried out in a research project at Lancaster University which has been looking at computer support for air traffic controllers [18]. As suggested, probably the most important lesson that ODP community needs to learn in this context is the necessity to first understand the nature of the problem (e.g. the sociality of work) and to exploit this knowledge in building effective computer support [18]. This activity naturally belongs to the ODP enterprise (and information) viewpoints and specification.

2.7.2 Psychology

Being a science which studies the behaviour of man, psychology can be helpful in provid-

ing a deeper understanding of the various aspects of individual-technology interactions. For example, its results can be used as an input in the design of educational software (exploit subject matter on how learning occurs) and the design of ergonomic human-machine interface (exploit its subject matter on perception). As a part of this thesis, we will use a particular psychological theory, Information Integration Theory (IIT) [1], as an ingredient of our QoS metrics framework. This theory provides a general framework for analysis of customer multi-attribute alternatives. It can be used to determine the model which most appropriately represents the psychological processes by which consumers combine different attributes of services. This theory, and its application to QoS metrics framework will be presented in chapter 4.

CHAPTER 3

Economic and Business Aspects of Open Distributed Systems

The previous chapter has illuminated broad categories of enterprise issues which arise in the context of ODSs. The focus of this chapter is confined to the *economic* and *business* characteristics of ODSs. We will outline those economic theories that we find relevant for ODSs and study how they can be related to different economic and business aspects of ODSs. In each of these sections we will present applications of these specific theories to relevant problems within the domain of ODSs.

After the introduction in section 3.1, we will outline the basic concepts of traditional micr-oeconomics (section 3.2), with the aim of emphasizing its inadequacies in addressing a large class of economic related characteristics of ODSs. The failure of traditional microeconomics to include quality of products or services as a variable in economic analysis will be first highlighted. This will be followed by an outline of Lancaster's theory of consumer demand (section 3.3), which regards quality as a highly differentiated commodity and includes this fact in the theory. We will exploit this theory to derive a general, service independent QoS framework and associated QoS metrics in chapter 4.

Traditional microeconomics assumes that the market is the coordinating mechanism for economic transactions. Since ODSs will be implemented within organisations and across markets, we look at transaction cost economics in order to extend the domain of economic transactions to organisations. Transaction cost economics (outlined in section 3.4) highlights issues which determine whether economic transactions should take place within a market or within organisations. We will utilise this theory at several points in the thesis, such as to analyse the operations of the RM-ODP trader component, in different settings (public, private or third-party owned,), section 3.11, as well as to gain more insight into the concept of contract (section 6.1).

Further, traditional microeconomics' assumption of perfect information about commodities is often not appropriate in the context of ODSs. We show why this assumption is not realistic by elucidating different reasons for uncertainty in service delivery in ODSs. We demonstrate how economic agency theory (outlined in section 3.5) can be applied to incorporate uncertainty in designing contracts between players in ODSs. This will be studied in more detail in chapter 5.

Additionally, it is beneficial to understand various types of organisational forms within which ODSs are to be implemented (e.g. within which the trader can operate). Different economic theories of organisation provide insight into these topics and will be outlined in section 3.6.

In addition to economic issues, relevant concepts from business practices are discussed as we believe that these also need to be considered when designing a commercially acceptable ODS architecture. Section 3.7 addresses the business topics of strategic management and their relevance for ODSs.

We will also attempt to provide an integrated view of the aforementioned economic theories and business concepts and their interrelationships. Since we argue that game theory can provide a unifying link between these (i.e. within the framework of game theory, the previous concepts from different economic theories can be formalised), the main concepts from game theory will be first described (section 3.8), followed by an integral picture of

these theories (section 3.9).

While in each of the above sections, we investigated how the corresponding economic theory can be applied to the pertinent resource issues in the context of ODSs, in section 3.10 we will inquire into the impact of ODSs on organisations and markets, which is an economic topic in its own right.

Finally, in section 3.11, we will illustrate how the ODP Trader component can be treated as an economic entity and how appropriate economic theories can be used to elucidate a number of economic concerns associated with this service.

3.1 Introduction

As a consequence of the changes that computing and communications industries are experiencing (as accented in chapter 2), it is envisaged that a number of new economic issues will arise within the scope of ODSs. This is augmented with the fact that in many ways emerging distributed systems resemble large economic systems (which we refer to as open information markets), within which new types of interactions between (commercial) partners will emanate. For example, a typical economic issue involves the analysis of critical factors which determine competitiveness of enterprises in such markets. These are not only *price*, but also *quality* and *choice* as stated in chapter 2. These factors are the central interest of the following main groups of players: *users*, *network providers, service providers*, and *regulatory bodies* (Fig. 3.1) [112]. On the demand side, users are increasingly becoming aware of the strategic importance of IT in achieving critical missions for their enterprises and are now more selective when making choices, while striving to capitalise on the capabilities of new IT. On the supply side, network and service providers and equipment manufacturers understand that they need to provide better quality and lower prices in order to respond to market needs and further extend or diversify potential markets. In order to assure greater market coverage, service providers increasingly analyse other competitors' moves and make strategic decisions accordingly.

It is worth emphasizing here an increasing attention to Quality of Service (QoS). This re-

sults from the fact that QoS is now regarded more than simply an engineering variable supplied by network and service providers (as was the case in the past). Rather, it is a parameter defined by users, based on their perception, and is thus an important economic variable of interactions between users and providers. Additionally, QoS represents a major element of competitiveness of providers.

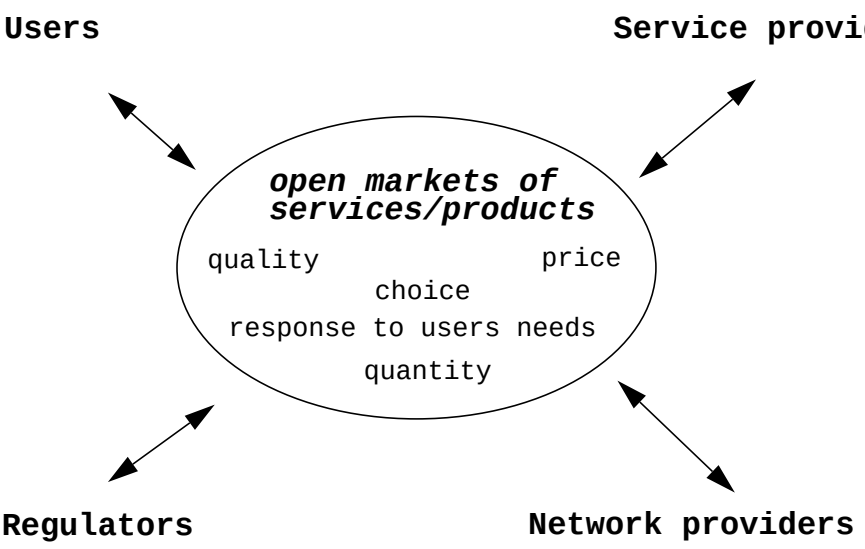

Figure 3.1 Main groups of players in an ODS

However, due to new competition variables (e.g. QoS) and other economic factors inherent to ODSs (which will be discussed in chapter 5), the pure competitive environment (and hence the perfect market) is not easily achievable. Consequently, traditional microeconomics cannot be used to address all of the ODS economic driven concerns. In other words, in addition to competition, there is a need for other types of institutional arrangements and organisational designs. One of them is *regulation* [137]. For example, regulatory bodies have important roles in the regulation of the telecommunications sector, which includes the licensing of competition, the authorisation of tariffs and monitoring of QoS [112]. Their aim is to ensure that the benefits of new technology can be evenly spread across society, which should increase overall social welfare.

Another common mechanism, used in situations where there is imperfect information

about the environment, is the provision of performance oriented rewards or *incentives*. For example, in the presence of externalities and *uncertainty* about the quality of commodities, services or rights exchanged [137] one needs to devise incentive schemes which would alleviate the undesired effects arising from the lack of perfect information. Since uncertainty can be a significant factor in ODSs (as will be elaborated in section 5.1), there is a need to address these issues too when dealing with open markets of information services. In the context of services in ODSs, a major source of uncertainty is related to the provision of agreed upon QoS levels. It will be shown (in section 5.3) how economic agency theory can be applied to a large class of services in ODSs. This theory can provide a suitable mechanism to analyse and model behaviour of economic actors (e.g. service providers and service consumers) in the presence of uncertainty. It can also be used to design contracts which would be efficient from both, users' and service providers' viewpoints.

The above economic and business issues ought to be appropriately considered in any ODS architecture if it strives to gain a wide commercial acceptance. For example, the fact that ODSs require interworking across different boundaries has implications for enterprise strategists, enterprise planners, designers of specifications and application programmers. These boundaries may be organisational (e.g different organisational units), economic (e.g different national economies, different organisations), social (e.g various social systems), political, cultural and technological [55]. Therefore, in ODSs, interaction rules between components cannot be implicit as is the case in many existing information systems[1]. They should be one ingredient of an ODS architecture, since such an architecture should provide a set of principles and rules which allow autonomous agents to have a common view on what is beyond their boundary and how to interact with other domains. This common set of principles and rules should facilitate exchange of information or services (in economics, such an exchange is called a transaction). For example, such rules can be included in *contracts* - which in the simplest legal terms is 'a promise enforceable at law' [44]. The incorporation of the legal aspects of contracts ensure the legal validity of the transactions between businesses. Another aspect of contracts which is of more relevance for service specification relates to the problem of *designing the contract*. This is im-

1. In conventional systems principles and rules which allow autonomous agents to interact with each other are implicit in protocols, naming conventions etc.

portant from the economic point of view since contracts arise as a result of efficiency-seeking behaviour in a world of uncertainty [114].

Our motivation for investigating relevant disciplines from economics comes from similarity of the above mentioned issues with the problem domain of economics. Namely, economics can be defined as a science of cooperation with regard to the utilisation of resources [137]. Economics is concerned with a major problem which can be formulated as the search for the optimal allocation of scarce resources. Economics has the purpose of *i)* explaining economic phenomena and predicting events (positive aspects) and *ii)* addressing the problem of how resources should be allocated (e.g. equitable distribution of wealth) and social utility theory (normative aspects). Depending on the problem domain, several different economic theories[2] which can be applied to relevant ODS issues are distinguished, as follows.

- *Traditional microeconomics* - inquires into the problems associated with markets.
- *Lancaster theory of consumer demand* - extends the traditional microeconomics with the notion of quality.
- *Transaction cost economics* - introduces the cost of transactions and explores how these costs determine whether the transactions will occur in the market or within the firm.
- *Agency theory* - provides guidelines for designing efficient contracts between economic actors in a world of uncertainty. It is a special type of problems which are the subset of a broader theory, *game theory.*
- Different *theories of organisation* - studies the different forms of organisations and looks at them as economic entities. This includes transaction cost economics, agency theory and behavioural theory of the firm.
- *Game theory* - provides a formal methodology for modelling of (cooperative and non-cooperative) interactions between rational decision makers, under different circumstances.

2. We note that modern microeconomic theory covers all these specific economic theories [75].

Hence, in order to be able to understand the nature of new resource problems and interactions arising in ODSs and to address these, we will be investigating the applicability of the aforementioned economic theories to the relevant concerns in the context of ODSs.

In addition, and from a more practical point of view, we argue that one needs to assure the integration of architectural concepts, structures and components which reflect business and commercial aspects of an ODS into the supporting architecture (e.g. QoS management, billing, accounting, contractual framework etc.). To this end one needs to incorporate concepts and principles commonly used in *business practices* into an ODS architecture, so that it can be widely accepted in commercial world.

To summarise, two important ODS objectives of economic relevance are:

- to make the numerous benefits of distributed systems (e.g. increased availability, performance, decentralised management) accessible to a wider user community
- to provide an effective sharing and an efficient utilisation of information services as well as computational, communication and other resources, across organisational boundaries.

With these prevailing goals, a number of new targets are set for ODSs. These include increased interworking between heterogeneous products and technologies, interoperation of different administration domains, and more open access to the public (as identified in section 2.3). However, in order to realise these targets, one faces new challenges. These can be roughly grouped in the following categories [96].

1. Provision of mechanisms which would accelerate users' confidence in accepting new characteristics of services in an open information market and thus alleviate uncertainty about the whole spectrum of issues to which the telecommunication and computing societies have not been previously exposed. This uncertainty arises from:
   a) the lack of users' previous experience with new services that are proliferating and evolving at a dramatic pace and thus uncertainty about what benefits can be gained from these
   b) the problem of succinctly expressing the semantics of services required

c) the problem of discovery of services in the global market place.

Some of these concerns (such as *a*) go beyond the scope of technology and also represent a new field of inquiry for other disciplines, e.g. psychology, consumer behaviour, economics and logic. Others (e.g. concerns *b* and *c*) represent new research topics for computer science. The issues from this category are beyond the scope of this thesis.

2. Developing a framework which would facilitate conceptualisation of the term QoS in ODSs and measuring it. This should be applicable to any service and should be based on user's definition of QoS. This will be the topic of chapter 4.

3. Solving a variety of new *resource problems* in ODSs which arise from inherent uncertainty of an ODS environment, but also from the characteristics of consumers and providers: there is a multitude of different consumers and providers having different requirements, preferences and information. This adds a new (to a great extent non-technical) dimension to the problem of efficient resource allocation and sharing in ODSs, which includes issues that have not been extensively studied in computing/ communication communities to date (and are indeed a major area of inquiry in modern economics).

   For example, different parties, in striving to achieve their own and often mutually conflicting objectives can take advantage of private information, behave opportunistically or take advantage of the actions of others. As a manifestation of these actions, an increased uncertainty of service delivery (QoS delivery[3]), can arise. This particularly refers to those information services which at a given point in time can be regarded as non-standard and which can involve significant technical complexity. Further, this uncertainty can in turn cause an uneven spread of the benefits of the underlying resources among interacting parties. The latter is a direct consequence of the twofold nature of QoS in an ODS. It has economic aspects since it is (next to quantity and price) the major element of competitiveness and an important element of contracts, as well as technological aspects since it describes requirements regarding physical resources in an ODS. In order to alleviate the 'unevenness', these potential behav-

3. Since our view (the reasons for which are given in section 5.4) is that uncertainty of service delivery is mainly related to its quality, we will use the terms service delivery and Quality of Service delivery interchangeably hereafter.

ioural patterns need to be taken into account when designing mechanisms for efficient allocation of resources among parties. This will be studied in chapter 5.

4. Identification, specification and design of *architectural components* to support these new features. Examples are concepts and entities within emerging management frameworks [133], as well as commercial contract support architectures, which should support users in expressing their business policies and requirements and thus, facilitate business transactions. These architectural aspects are discussed in chapter 6.

3.2 Traditional microeconomics: basic concepts

Traditional microeconomics[4] deals with the problem of coordination of economic decisions (of consumers and producers) via markets. A *market* is a collection of consumers and producers who interact, resulting in the possibility of exchange. Consumers can choose between a large number of products or services and make decisions about how much of these products or services they are to consume. Producers make decisions about quantities of products or services to be produced and the technology of production. Coordination between producers and consumers is achieved in markets via the price mechanism.

3.2.1 The theory of supply/demand

Supply and demand curves represent the relation between the price of a good and the quantity supplied and demanded (Fig.3.2). The supply curve shows how much producers are willing to sell for each price that they receive in the market. The demand curve shows how much consumers are willing to buy for each price per unit they pay.

The two curves intersect at the equilibrium or market-clearing point. At this price, quantity supplied and quantity demanded are just equal. The market mechanism is the tendency in a free market for the price to change until the equilibrium point is reached.

4. The material in this section is predominantly based on [115].

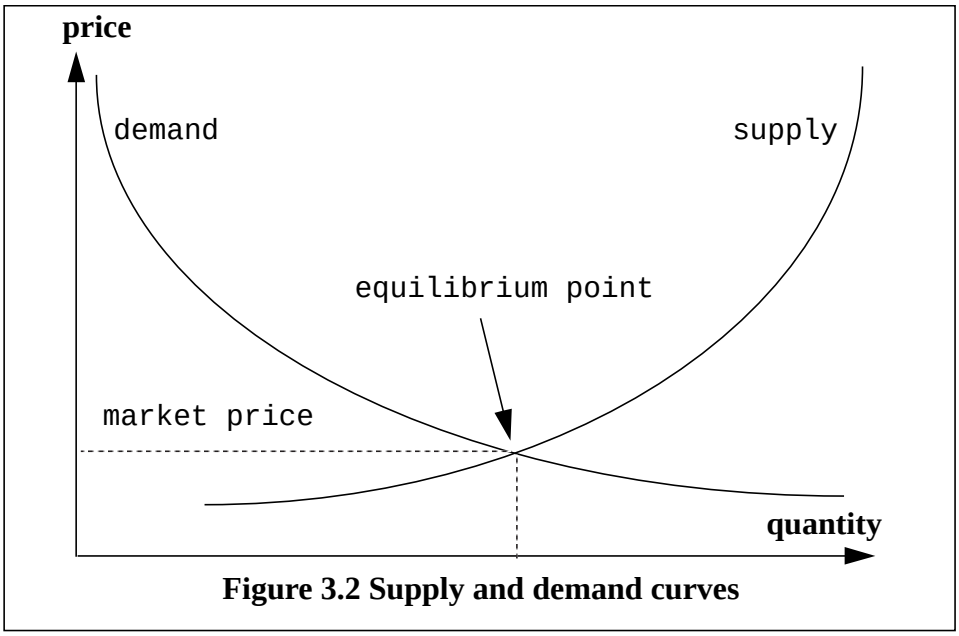

Figure 3.2 Supply and demand curves

The theory of demand

Traditional microeconomics assumes that people rank different collections of goods in order that reflect their preferences. This can be represented by a set of *indifference curves* (Fig. 3.3). For example, a user is indifferent between combination of products P_1 and P_2, But the user prefers all combinations represented by points on U_2, as compared to combinations of products on U_1.

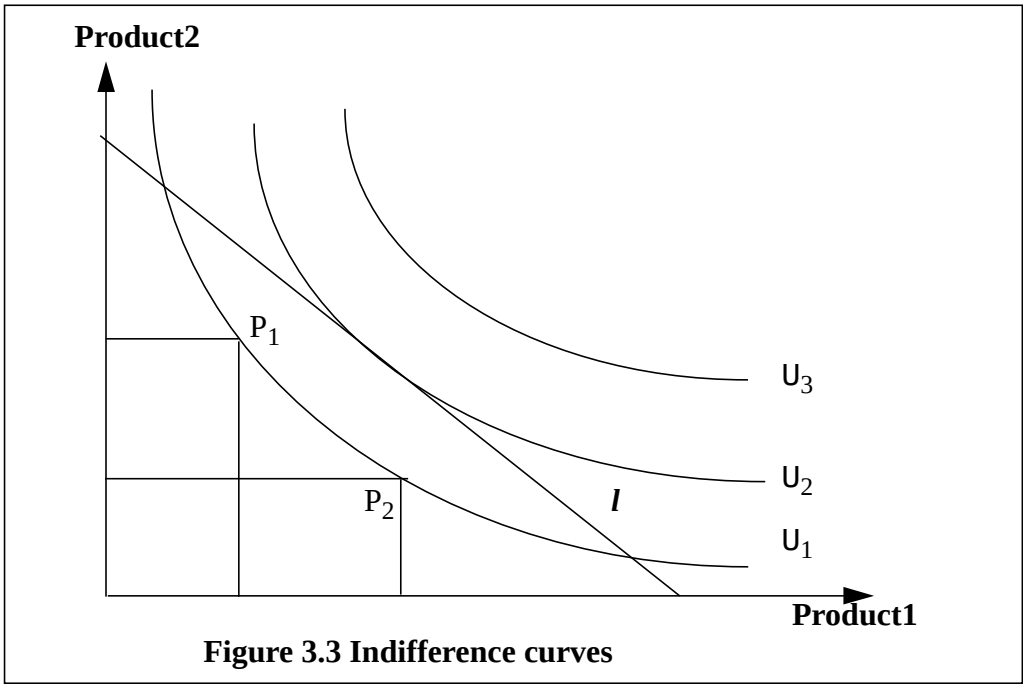

Figure 3.3 Indifference curves

However, the amount of products of type 1 and 2 that a consumer can buy depends not only on the preferences but also on his income or budget. The quantities of the two products that the consumer can buy are given by line *l*, called the budget line. If we assume that every consumer wants to maximise his level of *utility*, he will choose the point on this line, that gives him the highest level of utility, i.e. he chooses such indifference curve which is tangential with the budget line *l*. *Utility* is the level of satisfaction that a person gets from consuming a good or undertaking an activity. In economics it is used to summarise the preference ranking of customers. The *utility function* is obtained by attaching a number to each preference.These numbers are used for ordinal (not cardinal) ranking of utility levels. The utility function states the preference relations. Facing uncertain choice, individuals tend to maximize their *expected utility*. Expected utility is the sum of the utilities associated with all possible outcomes, weighted with the probability of these outcomes.

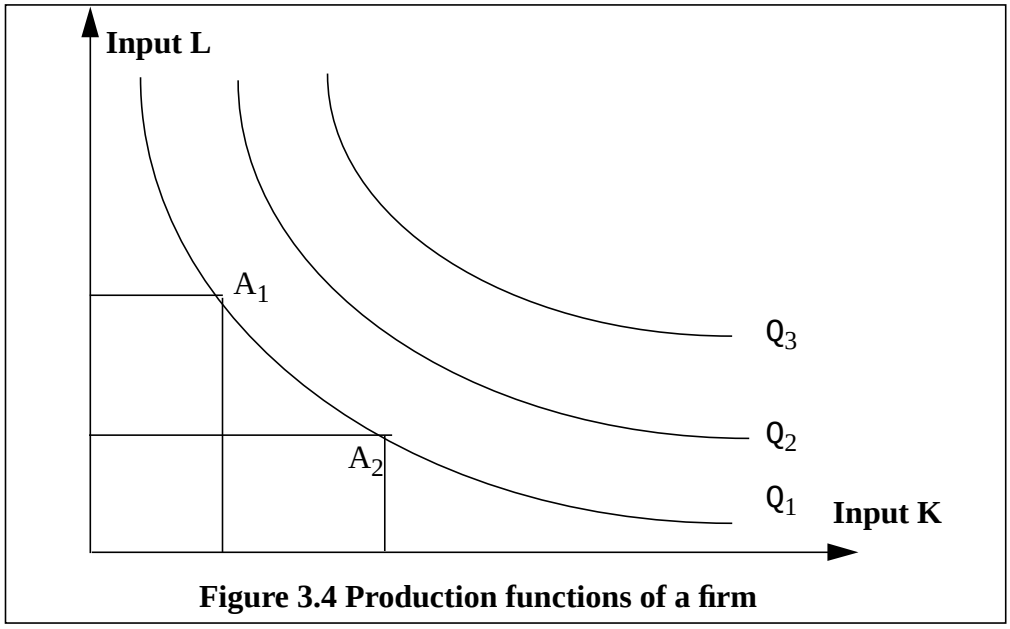

Figure 3.4 Production functions of a firm

The theory of production

In traditional microeconomics, the firm is described as an entity that maximizes an objective function, which describes the firm's goals. This objective function can be maximised within the constraint given to the firm by its *production function*. The production function describes the relationship between any combination of inputs and the maximum output a firm can produce with those inputs. For example, the production function with two inputs K (amount of capital) and L (the amount of labour) is depicted in Fig. 3.4. Each of the

curves represent all possible combinations that the firm can choose to produce a given quantity. In order to produce more output given the same amount of one input, the firm needs to increase another input variable. Thus, $Q_3 > Q_2 > Q_1$.

Now, in trying to answer questions about production, traditional microeconomics assumes that the firm's goal is profit (revenue minus cost) maximisation.

In maximising its production function, the firm takes into account whether a particular input can be changed or not. For example, it is assumed that in the *short-term* the firm cannot change all the production factors (inputs). Rather, it is assumed that only one of them can be changed and the firm should choose the amount of this input in order to maximise its output. This can be done by finding the first order derivative of the production function with respect to this variable. On the contrary, in the *long-term* it is assumed that all the input variables can be changed and thus the profit maximisation is now a problem with two (or more) decision variables.

3.2.2 Model of perfect competition: major assumptions

As has been shown so far, traditional microeconomics explains how the price mechanism achieves coordination between quantities demanded and quantities supplied. It is based on the following assumptions [40].

- There are large number of small producers and consumers. The implication of this assumption is that their decisions do not affect market price, and that they are price takers, whereby the market price is determined by the total industry demand and supply.
- There is free entry and exit of firms to and from the market.
- Each market is characterised by standardised products, i.e. it is irrelevant for consumers from which producers they buy these products.
- There is perfect information, i.e. everyone knows everything that is relevant for making decisions.

- Firms are viewed as holistic entities (there is in fact no difference between the concept of a single producer and a firm - they are both modelled as a production function with an objective, e.g. profit maximisation).
- Firms have a single objective (normally to maximise profits or their value on the stock market).
- Behaviour of producers and consumers is some kind of maximisation.

If all industries were adequately described by the model of perfect competition, we would live in a maximally efficient world [40]. The result of free competition would then lead an allocation of resources which is referred to as Pareto efficient (or Pareto optimal). *Pareto optimality* is the criterion used in economics to define the optimal allocation of resources. It essentially states that a set of allocations of resources to agents is Pareto-optimal if no agent can be better off (i.e. be given a better allocation of resources) without making an other agent worse off (i.e. to be assigned a worse allocation of resources) [83].

3.2.3 Limitations of traditional microeconomics

It is important to note that traditional microeconomics is inadequate in addressing the economic problems of markets in which there are different levels of qualities of products or services. One of the fundamental assumptions of traditional microeconomics is that firms and consumers are price-takers, buying and selling *homogeneous* commodities in well-defined marketplaces [139]. However, the emerging open information markets emphasize a new economic issue. Rather than being taken for granted, quality must now be regarded as a *highly differentiated commodity* [112]. Some of the traditional economic theories (e.g the theory of supply and demand, the theory of the single market price) do not hold in markets where a number of firms offer services of different quality at a range of different prices [139] [5]. Further inadequacy of microeconomics in the context of ODSs is reflected by its limitation in situations where there is imperfect information, e.g. due to uncertainty of service delivery, particularly with respect to QoS.

5. In such deregulated markets, price could have an additional function. It could convey information and affect behaviour, particularly in circumstances in which there is uncertainty at some point.

We will see later that all these issues arise in the context of ODSs and thus we will need to look at the applicability of institutional arrangements and organizations, such as performance-oriented rewards (incentives), and regulation (enforcing contracts, penalties if rules are violated), in addition to the traditional market-driven mechanisms (e.g. *competition* [55]).

3.2.4 Applications to ODSs

Some concepts from microeconomics have proved useful in designing resource allocation problems in the context of traditional distributed systems [47]. Since these systems could be characterised with perfect information, microeconomics algorithms have been useful in such circumstances (i.e. where there is perfect information regarding some variables, e.g. processor load, available bandwidth etc.). However, due to the assumptions which traditional microeconomics adopts and which are unrealistic in many situations characteristic of ODSs, its applicability to the analysis of services in ODSs has limitations. In fact, traditional microeconomics can be applied in situations where there is a standardised information service known to everyone (in all its aspects) and where there are many competing providers of such services. When services become richer (in terms of the number of characteristics), applying microeconomic concepts becomes harder and often inadequate (as discussed in the previous section) and service price is not the only mechanism for achieving coordination between consumers and suppliers. This is an important fact which should be recognised by those who are dealing with related economic-related enterprise issues.

3.3 Lancaster's theory of consumer demand

In order to address the inadequacies of traditional microeconomics regarding different levels of QoS and the implications, Lancaster has extended the conventional theory of consumer demand [78]. According to Lancaster's more general theory of consumer demand, products or services[6] are viewed as bundles of characteristics and consumers de-

rive utility from the characteristics embodied in services rather than from services per se. Utility orderings are assumed to rank collection of characteristics and only rank services indirectly through characteristics that the services posses. Characteristics are defined as ‘those objective properties of things that are relevant to choice by people’. Further, for analytical purposes, individual characteristics may be dis-aggregated into sub-characteristics and several characteristics may be aggregated into an aspect of service quality [6].

In fact, the theory is based on two fundamental propositions, whereby it is possible to clearly distinguish:

- the relationship of *services* and their *characteristics*
- the relationship of *characteristics* and *people*.

The first proposition states that all goods possess *objective* characteristics relevant to the choices people make between different collections of services. The second proposition states that individuals differ in their *subjective* reactions to different characteristics, i.e. it is the characteristics of services, not the services themselves which people are interested in [78]. Therefore, services can be viewed as bundles of characteristics and *quality* refers to the quantities of individual characteristics (Fig. 3.5).

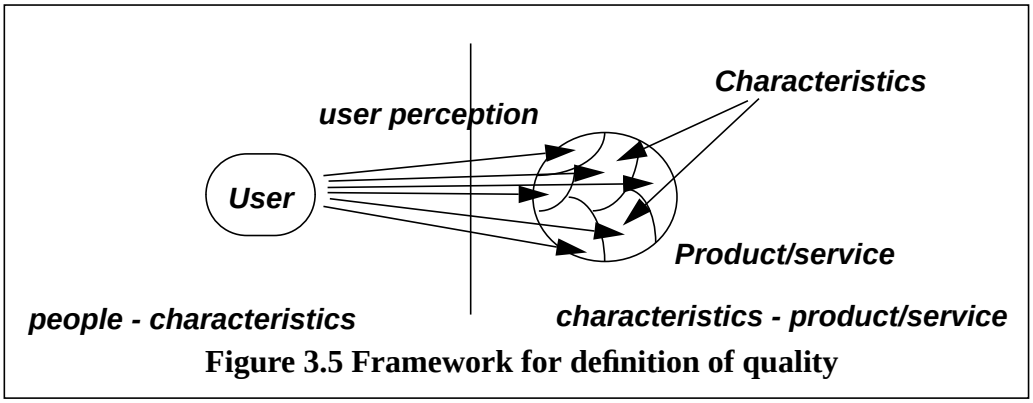

Figure 3.5 Framework for definition of quality

3.3.1 Applications to ODSs

Lancaster’s theory of consumer demand can be used as a paradigm to solve various qual-

6. Since our interest is in quality of *services*, we will use term services hereafter.

ity related problems arising in the context of ODSs. We will use it as a starting point to facilitate QoS conceptualisation and develop a comprehensive QoS metrics framework (on a user-driven, application-independent basis), as will be discussed at length in chapter 4.

3.4 Transaction cost economics

traditional microeconomics, with its emphasis on the study of economic transactions taking place at markets, appears to be inadequate in answering another economic-related questions: why all transactions are not carried out in the market place and why so many other different organisational forms exist? Answers to these questions can be found in *transaction cost economics*, an economic theory which has emerged in recent decades [151].

3.4.1 Foundations

Transaction cost economics recognises the fact that in addition to traditional production costs, *transaction costs* should also be taken into account in an economic analysis of commercial interactions. These include both costs of market transactions and costs of internal transactions. Based on this premise, Oliver Williamson has developed a theory, which can serve as a framework for comparison of the costs associated with market transactions and internal transaction [151]. He explains that firms exist because in some cases the costs of internal transactions are lower than the costs of market transactions.

In developing the theory, Williamson assumes two characteristics of human beings: bounded rationality and opportunistic behaviour.

Bounded rationality refers to the limitations of humans to formulate and solve complex problems[7]. Yet, many situations in economics are characterised by high levels of com-

7. The example of chess is given, in which players behave rationally, but their capacity to evaluate fully consequences of all possible decisions is limited.

plexity. In addition, there is often some *uncertainty* related to the information available when making economic decisions. It is the combination of uncertainty and/or complexity and bounded rationality that leads to transaction costs. To illustrate this, two examples from [40] can be used. In the first example, it is argued that buying petrol for a car does not pose much complexity or uncertainty and thus one does not need to sign a *contract* to do this (since petrol is a standard product the price is sufficient information for this market transaction). In other words, as a result of perfect information about goods, some markets are close to ideal markets, e.g. markets for raw materials, stock markets etc. In the second example however, it is shown that a government which wants to buy a new weapon systems faces much more uncertainty and complexity in considering various circumstances when designing contract (in this case, it is quite *costly* to write a contract).

Another characteristic of human beings which needs to be taken into account when looking at interactions between economic actors is *opportunism*. Williamson does not assume that everyone behaves opportunistically, but that it is possible that some people (sometimes) may behave that way. A typical example is a dealer in second-hand cars: buyers simply do not rule out possibility of the dealer's opportunistic behaviour. As a result the buyers are willing to spend some money to reduce the effects of such a behaviour by having the car inspected or drawing up contracts. Opportunistic behaviour can occur *ex ante* (i.e. before a deal is made) or *ex post* (i.e. after a deal is made). It is important to note that in cases where there are many sellers, their opportunistic behaviour is reduced, due to competition (e.g a dealer knows that his reputation, and thus his competitiveness will be damaged, and he reduces the opportunistic behaviour, so that a buyer can save on inspection costs) [40]. Hence, opportunism appears to be a problem only if there is a small number of trading partners. It is interesting to mention the so called *fundamental transformation* circumstances which arise when an original situation of large numbers exchange is transformed into a situation of small numbers exchange, owing to 'learning by doing'. For example, a supplier of a car component which has a contract with car manufacturer learns how to produce the component more efficiently, and by doing so it becomes a monopolist in this market [40].

Therefore, the transaction cost economics framework includes two human factors (bound-

ed rationality and opportunism) and two environmental factors (uncertainty/complexity and ‘small numbers’ exchange), as shown in Fig. 3.6 [40]. Human factors, coupled with environmental factors lead to transaction costs.

.

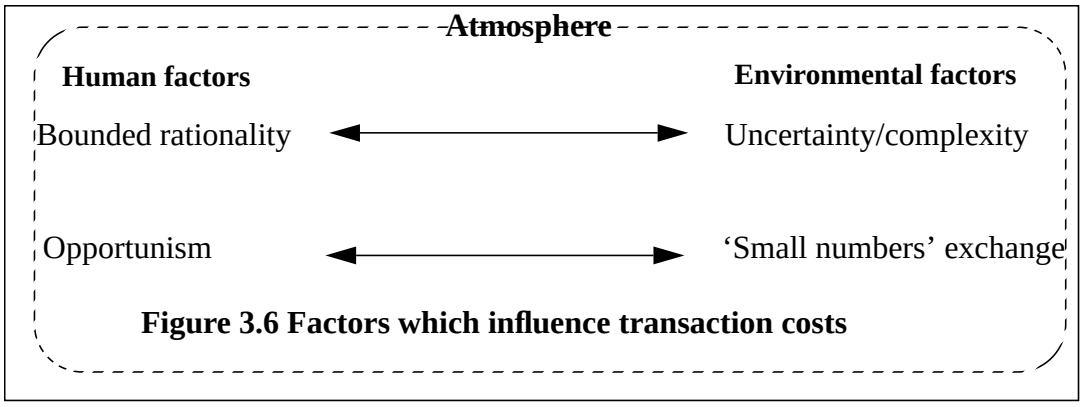

Figure 3.6 Factors which influence transaction costs

The mode of transaction (in terms of whether it is carried out in firms or across markets) is determined by minimisation of the sum of production and transaction costs. Additionally, there is a third factor, called *atmosphere*, which basically represents the fact that participants in a transaction may value particular mode of interacting (e.g if people prefer to work as self-employees, they will be willing to trade this only if the higher income of working as an employee compensates for the loss of ‘atmosphere’ [40]).

3.4.2 Cost factors of transactions

Transaction costs depend on several critical aspects of transactions. These are as follows [151].

1. *Asset specificity* reflects the degree to which a transaction requires transaction specific assets; this is an asset which cannot be redeployed to an alternative use without significant reduction in its value. For example, if asset specificity is high, one expects that transactions will be carried out within organisations rather than across markets. In this case, the more frequent transactions - the more easily recovered fixed costs associated with setting up of a specialised governance structure (vertically integrated firm).
2. *Uncertainty/complexity* associated with transactions influences their cost, as already mentioned in the previous subsection.

3. The *frequency* of transaction. If a certain transaction is frequent, then the costs of setting up a specialised governance structure are more easily recoverable. This is particularly the case for asset specific transactions.

The relationship between characteristics of transactions and the most appropriate governance structure will be elaborated in subsection 6.2.1 in the context of an analysis of uncertainty associated with a class of services in an ODS.

3.4.3 Applications to ODSs

Owing to its emphasis on the transaction cost analysis, transaction cost economics can provide a framework for enterprise planers for decision making regarding different resource deployment alternatives. An example of such an analysis, with respect to the choice of trading service to be used, will be given in section 3.11.

Once the analysis is carried out, both system and application designers can use this knowledge as an initial guideline about organisational structure, which should be mapped to the most appropriate system and application solutions. For example, system designers can use this knowledge for optimal structuring of the distributed systems. They can make decisions about the optimal allocation of processing power, within the organisation and at organisational boundaries, according to current organisational needs (e.g. is it more economical to have a 'fully distributed system', consisting of workstations of equal processing power, or mainframe based system, or some combination of these). This knowledge can also be a useful tool for application designers, who should design applications that most naturally reflect the structure of the organisation.

Transaction cost economics can also serve as a starting framework for understanding the future organisational evolution, so that ODSs can gracefully evolve (and possibly influence the evolution of the organisation), as the organisation grows and changes. Such an investigation is outlined in section 3.10.

Further, transaction cost economics can be applied to the economic analysis of electronic

interactions between firms. This is of particular relevance for ODSs, as they will promote inter-organisational business dealings. An example of how this can be done is presented in [156]. In the following, we outline the main findings from this research.

Example: an economic analysis of electronic inter-organisational contracts

In this study, transaction cost economics has been used to first develop and then empirically test a model of determinants of the degree of electronic integration in one specific commercial sector, the insurance industry. The study has identified several critical factors of the integration between a principal and an agency, of which business process asset specificity and trust have received strong empirical support.

Business process asset specificity comprises human asset specificity (specialised labour skills and experience) and procedural asset specificity (workflow and procedures that are customised for this particular relationship). These IT specific investments are used to provide enhanced business opportunities for restructuring business relationships with chosen business partners. It has been hypothesized, and with empirical tests proved that ‘asset specificity is positively related to the degree of electronic integration’. This means that the implementation of a dedicated, inter-organisational information system creates non-redeployable specific assets due to which it is costly to switch to alternative principals, so that the agency should channel more business to interface the principal.

Trust is enhanced through greater ease of communication between the carrier and the agency via dedicated inter-organisational IT system. Empirical tests have also proven that ‘trust will be positively related to the degree of electronic integration’.

The importance of these findings is in empirical justification of determinants of electronic integration. This can serve as a basis for future empirical assessments from a transaction cost perspective: those relating to the role of IT, in particular the capabilities of ODSs, in influencing business relationships. For example, the asset specificity has emerged as a significant predictor of the intermediate (i.e. between market and hierarchy) governance mode in a service sector such as insurance. These findings have important implications

for managers: they can serve as guidelines for the formulation of an electronic integration strategy (e.g. proprietary versus common systems).

3.5 Agency theory

Agency theory in general terms focuses on a multitude of problems which characterise relationships between one or more players called *principals* and one or more other players called *agents*, who make decisions for, or act on behalf of principals. These relationships, frequently referred to as *agency relationships*, acquire interest if there is *uncertainty* at some point [4], emanating from information asymmetry and environmental risk, Fig. 3.7. Agency relationships exist within organisations (e.g. between an owner and a manager) but also across markets (e.g. between a buyer and a supplier) [43].

3.5.1 Information asymmetry and environmental risk

Information asymmetry refers to a situation in which at least one of the players possess private information which gives some kind of an informational advantage. There are two broad categories of information asymmetry which are of interest for agency theory. The first one refers to situations which occur *after* the contract between a principal and an agent is set up (ex-post), whereby the agent, in order to pursue his own objectives, can act in a way which is not optimal for the principal (but the principal cannot observe it). This type of situation is called *hidden action* (or moral hazard). Another type of problem is called *hidden information* (or adverse selection); it arises when the agent has knowledge not shared with the principal, and bases his decisions on that knowledge. Hence, it refers to circumstances *before* the contract is signed (ex-ante). The terms moral hazard and adverse selection come from the insurance industry. As a result of information asymmetry, the agency problems can arise: when the principal and agent have conflicting goals and when it is difficult (or expensive) for the principal to verify what the agent is actually doing [43].

Environmental risk emanates from the unpredictability of the environment which sepa-

rates principal and agent (state of Nature), Fig. 3.7. The presence of the environmental risk introduces not only inability to predict, but also risk that must be borne by someone: principal and agent are normally interested in optimal risk sharing, since they can have different attitudes towards risk (and this in turn can determine their different actions).

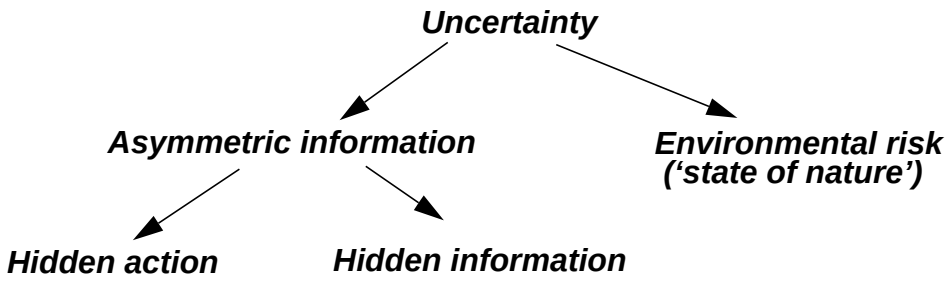

Figure 3.7 Various types of uncertainty within the agency theory scope

There are two streams of agency theory study, namely the positivist theory and the principal-agent research. The former aims at explaining how different governance structures solve the agency problems, e.g. how the organisations can exist in spite of agency problems (as will be discussed in subsection 3.6.2). The principal agent model is more formal: it focuses on the design of the most efficient contracts, under various levels of risk aversion, information and opportunistic behaviour. This will be studied in detail in chapter 5, in the context of service provision in ODSs.

The central point of the principal agent model is to find a fee which optimally trades off the benefits of risk-sharing with the cost of providing incentives to the agent (incentives are introduced to cope with asymmetric information problem) [119]. This fee, which the principal pays as a reward to the agent for his effort, is a variable which needs to be determined and could be a complicated functional relationship.

It is now widely recognised that agency relationships dominate many economic activities, within organisations and across markets, and we conjecture that they can also take place within ODSs (as will be explained in detail in chapter 5).

3.5.2 Applications to ODSs

traditional microeconomics' assumption of perfect information is unrealistic in the context of ODSs. We have pointed out (in section 3.1) that agency theory can be applied to design contracts which would cover service delivery in the presence of uncertainty. Sources of uncertainty which are related to service delivery in present information systems can be found mostly within the domain of different technical variables such as reliability, availability and responsiveness.

However, these sources of uncertainty are augmented in ODSs. The process of identification of service providers, contract negotiation and subsequent interaction among users and service providers can bring additional uncertainty. This can be due to *unpredictability* of the environment (moves of Nature), but also to the limited capabilities of players (the *bounded rationality* constraint), their opportunistic behaviour, and to the possession of private information (as mentioned in section 3.1). It is likely that one or more of these factors prevails in the process of service provision in open systems, directly impacting players' benefits in such systems. Consequently, a combination of technological and non-technological issues which account for other types of uncertainty should be taken into account: those which arise as a result of openness, distance, separation between users and service providers, and unobservability of different players' actions during their cooperation.

Given perfect information about price (quite a realistic assumption), the major source of uncertainty is the *QoS* that the SR obtains as a result of the actions of the SP. One therefore needs to consider these uncertainties when designing contracts between players; this indeed being a non-trivial resource allocation problem. These issues will be discussed in more detail in Chapter 5, which discusses different sources of uncertainty in ODSs and how agency theory can be applied to alleviate uncertainty in ODSs.

3.6 Economic theories of organisations

traditional microeconomics assumes that price carries all the information necessary to co-

ordinate the activities of economic actors. While this holds for markets, it is evident from practice that there are a number of other coordination mechanisms which are used within *organisations*, and which define *structure* of organisations (firms). In this section, we discuss different structures of organisations. We begin with the transaction cost economics explanation of organisations, which can be applied to explain the existence of different organisational forms. Some other theories of the firm will also be presented, e.g. a view of a firm as a network of different contractual relationships (based on agency theory concepts) and the behavioural theory of the firm.

The behavioural aspect of organisations, particularly in terms of their interactions with other organisations is explained in section 3.7.

3.6.1 Transaction cost economics explanation

The most simple organisation is the *peer group*, a group of people working together without hierarchy, whereby the prevailing coordination mechanism is mutual adjustment[8]. There are several economic reasons why people form peer groups [40].

- Gaining of economies of scale. They may be obtained by having a joint ownership on expensive resources (e.g. farmers of harvesting machines, dentists of X-ray equipment), by economising on information gathering etc.
- Obtaining risk-bearing advantages by pooling the risk (e.g consultants trying to insure against the situation of not having assignments for an extended period). It is worth noting that peer groups serve as a good mechanism to alleviate hidden information and hidden action types of behaviour - since normally colleagues can better observe the effort of other colleagues as well as screen potential partners better than an insurance company. Thus, the cost of transactions is reduced if these transactions are carried out within the peer group than across the market.

8. In his well known reference [103], Mintzberg identifies six organisational coordinating mechanisms: mutual adjustment, direct supervision, standardisation of work processes, outputs, skills and norms.

- Enjoying associational gains which may be offered by peer groups, i.e. an individual may be more productive working as a member of partnership than when working independently.

However, the problem of the shirking can arise in large peer groups. In such situations, some members may be elected as managing partners - so that a peer group grows into another type of organisation - a *simple hierarchy*.

A simple hierarchy is a group of workers with a boss, who has the right to adjust wages, alter the structure of group, issue order etc. In this case the prevailing coordinating mechanism is *direct supervision*. The advantages of simple hierarchies over peer groups are in economies of communication and decision making (in peer group each member must communicate with all other members, while in hierarchy only with the boss) and in the capability to better monitor productivity.

More complex type of organisations are multi-stage hierarchies, such as *U-form* and *M-form enterprises*.

In a very large group, a single manager can no longer coordinate the work of team members (due to the bounded rationality) - in which case several, specialised managers are normally nominated. These managers' work is coordinated by a general manager. This organisational form is typical for medium size firms and is referred to as the unitary or U-form enterprise. However, once a U-enterprise attempts to diversify and produce other products, this forms appears to be inadequate. Firstly, in a large U-firm, there are several layers of management, which usually leads to a loss of information as it is transmitted (again result of bounded rationality) and ultimately to cumulative control loss. Secondly, with the growth of the U-firm enterprise, the character of strategic decision-making changes, since the managers of functional departments may be interested more in furthering the interests of their departments than in furthering the overall goals of enterprise. This is the result of a combination of bounded rationality and incapability to observe (leading to a hidden action type of situation). Consequently, it is very difficult to translate the overall goals of the enterprise into operational sub-goals of functional departments, since con-

flict of interests between functional departments will exist (due to scarcity of resources). The solution to these two problems is the M-form enterprise.

The M-form is divided at the top level into several quasi-autonomous operating divisions, usually along product lines, and the top management is assisted by a general office, which performs both advisory and auditing functions. The general office is concerned with strategic decisions, including the allocation of resources among the operating divisions. The problem of bounded rationality is alleviated in the M-form enterprise because the information is transferred less often. Opportunism is reduced because the goal congruence is easier to obtain - it is easier to translate an overall company goal (e.g profit maximisation) into operational subgoals per division (profit maximisation for each division) [40].

As M-form enterprises grow, it becomes harder to draw the boundary among organisations and markets. For example, there may be internal markets within the M-form enterprise for intermediate goods or services, labour and capital. Further, internal and external markets can be fully separated, partly separated or not separated at all. On the other hand, (external) markets can be regulated (by government) or by other regulatory bodies which specify rules, with which various market parties must comply [40]. All this suggests that there is always a mixture of market and organisational co-ordination mechanisms and that the combination of these polar type of coordinating mechanisms is a common fact of life.

It is worth highlighting another trend which has emerged in recent years - the trend towards *network organisations* [28]. They are emerging as a response to a need for more flexibility and sustained adaptability to an extremely competitive and turbulent environment, marked by increasing globalisation, technology change and technology transfer [28]. In such an organisation, separate firms, each retaining its own authority in major budgeting and pricing matters, function as integral parts of a greater organisation coordinated by a core firm. The problem of coordination in network organisation differs in major ways from coordination within markets or organisations. The major coordination mechanism is *reputation*, which characterises participants' behaviour in a network [28]: establishing and cultivating ongoing relationships among a selected group of companies is important for coordination in networks.

3.6.2 Agency theory explanation

According to the agency theory explanation of a firm (developed by Fama and Jensen [46]), firms can be seen as consisting of a set of agency relationships between self-interested parties (who are bound together in a *nexus of contracts*). As discussed in the previous section, when decision making is delegated to agents, it cannot be guaranteed that their decisions are aligned with the interests of the principal. Agency theory of the firm tries to explain how a firm is a viable form of economic organisation, even in the presence of agency problems.

Accordingly, the theory argues that there are different informational mechanisms which can limit an agent's self-serving behaviour [43]. Examples of such mechanisms are specially designed contracts such as outcome-based contracts which provide incentives for the agent to act according to the principal's interest, as well as information systems[9] which can inform the principal about what the agent is actually doing.

3.6.3 Behavioural theory of the firm

Behavioural theory of the firm addresses the decision making processes of firms, for example, decisions on price, output, advertising, investments etc. It was developed by March, Simon and Cyert [37], [89]. This theory departs from traditional microeconomics treatment of firms (as holistic entities with the objective to maximise profit), by viewing the firm as a coalition of participants each with their own objectives. Such a coalition need not have maximisation of profit as its sole objective. This is especially true for larger firms, whereby the decision-making processes involve two or more individuals (and not only the owner as in small firms).

In a typical medium to large firm structure, one can identify different parties, such as em-

9. In a broad context, e.g. reporting procedures, board of directors and an additional management layer. IT systems can also play a significant role in this, as will be discussed in section 3.10.

ployees, investors, suppliers, distributors and consumers [40]. According to the behavioural theory, each of these parties is interested in participating in the firm as long as the inducements offered to them are greater than the contributions required. The inducements are measured in terms of each party's values and in terms of the other alternatives available. For example, for employees, inducements include not only monetary payments but also non-monetary benefits such as self-fulfilment and job satisfaction. Contributions consist of labour hours, innovations and other non-time contributions.

With these assumptions as a starting premise, one finds the major difference between behavioural theory and traditional microeconomics: the assumption that all parties have perfect information about other alternatives. For instance, according to microeconomics, employees know exactly what they can earn elsewhere and, as soon as they are presented with the opportunity, they will leave and take another job [40]. Behaviour theory of the firm however suggests a more realistic assumption: the participants have their individual *aspiration levels* (e.g employee's aspiration level for his/her wage) which are adjusted slowly, as a result of other exogenous factors. Thus, if an employee hears that another company pays more for the same work, her/his aspiration level will adjust upwards. If, at some stage, the gap between aspiration level and actual payment becomes large, an employee will start looking for another job. However, this process is not instantaneous (as microeconomics proposes), and becomes more complicated because the inducements include non-monetary benefits, as mentioned above (due to which it may be even harder to obtain the right information). Another example can be a consumer who continues to buy from the same producer as long as the price is less than his aspiration level. If he obtains more information about lower prices from other producers, he will slowly adjust his aspiration level downwards. Similarly, consumers maintain an aspiration level about the quality of products, and the same reasoning about the adjustment of these levels will apply.

Therefore, due to incomplete (or even lack of) information, behavioural theory suggests that there is no perfect competition. In fact, it takes the competitive nature of markets as given, but focuses on the process of decision-making within the firm.

When analysing the firm's process of *setting goal(s)*, behavioural theory recognises that

the goals of the firm are arrived at through a bargaining process, characterised by the different bargaining powers of participants. This depends upon how unique a contribution a participant can offer to the coalition. Consequently, a real firm ordinarily has no single goal or clearly defined objective function. Only individual participants have goals expressed in terms of their aspiration levels.

In order to reduce the effects of bounded rationality, a firm's overall goal, e.g profit maximisation, should be translated into several *operational subgoals* over various departments (e.g production and sales departments). It is worth noting that (due to scarcity of resources) it is very difficult and often impossible to specify operational subgoals which are consistent with profit maximisation and which do not lead to conflicts between functional departments [40]. In behavioural theory, it is assumed that operational subgoals are specified and accompanied with the associated aspiration levels of managers - they will be rewarded if they at least reach these targets.

The process of goal definition is the first step in describing a firm's decision making processes. The second step is to describe how firms form expectations upon which the decision processes are based. Behavioural theory recognises the facts that decision-makers within the firm may have different information and may also derive different expectations from the same information.

The third step is to describe the process of organisational choice [40]. While traditional microeconomics assumes maximising behaviour of firms, behavioural theory assumes that firms make rough estimates of the consequences of their decision making. They do not maximise in the sense of evaluating all possible proposals for new products at the same time, because it is very often too costly to do so (both in time and resources). Moreover, due to the bounded rationality, it is often impossible to make all the calculations necessary to compare all the alternatives. Hence, in behavioural theory it is assumed that firms satisfy rather than maximise. This assumption is also in agreement with the view that the firm represents a coalition of participants with different objectives. Decision making processes in such circumstances are characterised by the fact that a decision about an alternative is accepted as soon as it meets the aspiration levels of all coalition members.

3.6.4 Applications to ODSs

It is now recognised that a major problem with the design of distributed systems that serve some organisational need is that the structure of the information system cannot be separated from the structure of the organisations it serves [61]. It is known that too often investments made in technology without considering organisational structure have failed. It is thus clear why one needs to understand organisational concepts, such as different organisational forms, reasons why they exist, as well as the organisational concepts needed to model organisations and changes that an information system must provide. Hence, an understanding of these concepts helps in designing a system which is 'best fit for purpose' Consequently, recent developments towards understanding interrelationships between new technologies and organisation business processes are taking an increasingly important role in the study of organisations and also in attempts to incorporate such issues into an ODS architecture. For example, new types of interactions between firms, such as those in network organisations need to be appropriately understood, so that an ODS architecture can be structured in a way that would best support key procedures in network organisations at all levels, as will be discussed in subsection 3.10.1.

3.7 Strategic management and other business issues

In section 3.4, we presented major factors which determine whether economic transactions take place at markets or within organisations. We also surveyed different organisational forms and found a big variety of such forms (in the previous section). Hence, our focus was on the explanations of why economic organisations exist and what are their different *structures*. However, not much attention has been given to the *behaviour* of organisations, which includes issues such as interactions with other organisations (competitors) at markets and adaption to external environmental factors. This is another area which has become a major subject of economics as well as business field of strategic management. Moreover, it can be said that today a substantial portion of the behaviour of organisations is the result of *strategic planning* and that to a large extent the results experienced by organisations can be explained by the quality of that planning and the basic economic conditions under which organisations operate [114]. To gain more insight into the behaviour

of organisations, we will discuss the relationship between economics and strategic management, followed by a description of major concerns of strategic planning and its positioning with regard to ODSs in the following.

3.7.1 Contribution of economic theories

Strategic management can be divided into two major thrusts: process and content components [40]. The former aims to help practitioners with the *process* of strategic planning, i.e. the formulation of a firm's strategy, taking into account the forces contributing to the strategic plan (Fig. 3.8). The economic contributions to strategic management focus on issues of *content*, i.e. on actual information firms need and the actual choices firms have to make in formulating their strategy.

It is recognised that an understanding of economic and managerial principles can make a striking difference in the quality of an organisation's strategic planning and the performance of the organisation [114]. The contribution of economics to strategic management can be inferred from a definition of strategic planning[10], i.e. "an organisation's strategic plan is its plan for allocation of its resources" [114]. More specifically, contributions from industrial organisation (explanation of different firms' performance at markets), transaction cost economics (explanation of contractual relationships) and game theory (forecasting and analysing different interactions among organisations) should be mentioned.

3.7.2 Factors contributing to the strategic plan

There are several factors which contribute to the process of strategic planning, Fig.3.8 [114].

1. A set of the organisation's *internal characteristics*. These include elements such as the present structure of the organisation, goals (e.g. short-run profits, growth) and current assets. These characteristics will constrain the organisation's set of feasible

10. *Planning* is a way to articulate a strategy for dealing with change in the environment and for creating change in that environment. A *strategy* is a commitment to undertake one set of actions rather than another.

options for new directions.

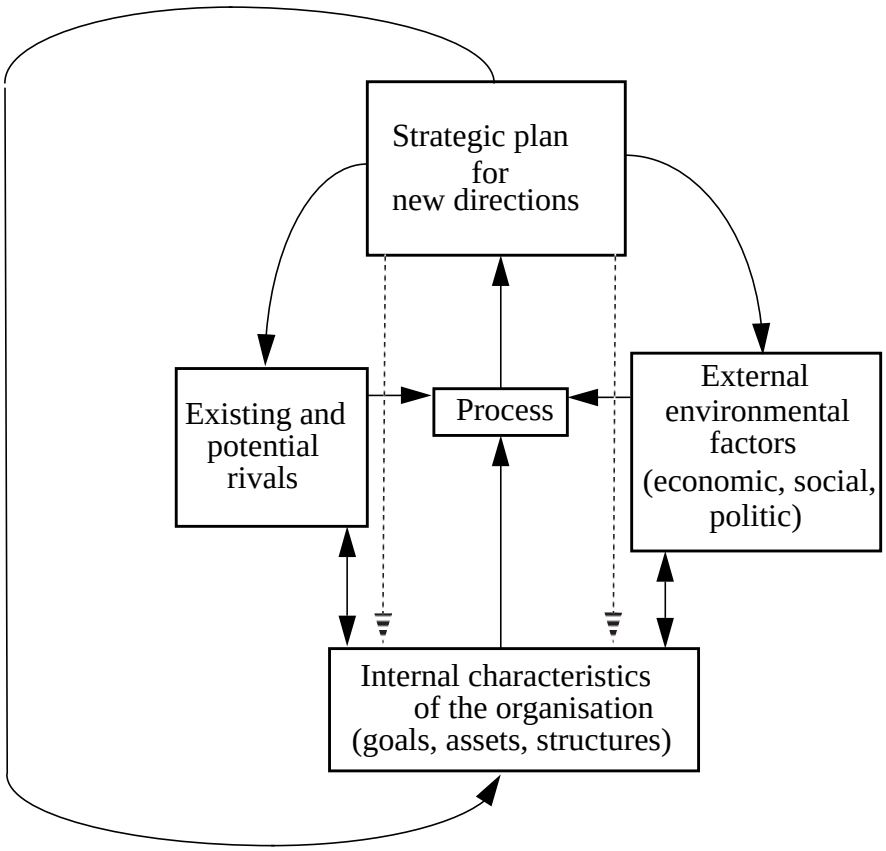

Figure 3.8 Factors contributing to the process of strategic planning

2. The organisation's *external environment*, which includes elements such as the laws and customs of society. These elements determine whether certain strategies can be adopted or not.
3. Existing and potential *rivals*. An organisation needs to analyse moves and counter-moves of other organisations while defining its own plan of actions. Game theory is helpful in developing an understanding of the variety of organisational interactions at markets. It is especially useful when one wants to trace the likely reactions of the organisation to the competitive moves of other organisations.

3.7.3 Process of strategic planning

Based on these three ingredients, the organisation can begin to define a plan to move in a

new direction, i.e. the process of strategic planning. This process varies considerably across organisations. A typical such process consists of a logical sequence of steps such as the one depicted in Fig. 3.9 [40]. The analysis of the environment (step 2) determines the intensity of competition and includes factors such as the number and size of firms, the level of product differentiation, the number and size distribution of buyers and suppliers as well as barriers to entry for new entrants. An assessment of the company's strengths and weaknesses relative to its competitors (step 3) belongs to the activity of competitor analysis. This involves an analysis of the firm's choices with regard to the strategic dimensions of that particular industry (i.e. key variables along which firms in the same industry differ, such as pricing strategy, advertising levels, degree of vertical integration), as well as strategic groups (i.e. groups of firms following similar strategic dimension). Step 4 includes formulation of static competitive strategy (two important competitive strategies are cost leadership and product differentiation) and dynamic strategy (which analyses moves and counter-moves of competitors). Corporate strategy (step 5) attempts to answer the question of where a firm should compete (e.g. in which markets and in which countries).

One objective of a strategic planning process is improving organisational response to a changing environment. Another objective is to map a vision for a new directions onto the organisation itself. An important element of this process is to design mechanisms for the control function within the organisation. This is particularly important for large organisations in which complexity and uncertainty may be prevalent factors.

The economic contributions to the process of strategic planning are made within the steps 2, 3, 4 and 5 (as depicted by the shaded areas in Fig.3.9).

3.7.4 Applications to ODSs

It is now recognised that IT has an important role as a strategic weapon to enhance organisational competitiveness. The advances in client-server computing (and more generally ODSs, as these mature) need to be taken into account as an important ingredient in an organisation's competitive strategy, since they can provide new sources of advantage for

business operations. These advances can be seen as a fundamental enabler in creating and maintaining a flexible business network of inter-organisational arrangements such as joint ventures, alliances and partnerships and long-term contracts. This represents an advance from the past role of IT, which was merely to increase the efficiency of organisations [148]. However, it is important to note that the benefits from deployment of ODSs into organisational business operations will be marginal if only superimposed on existing organisational structures and procedures. Hence, when considering the potential benefits of new technologies, one should also consider certain changes in organisational strategies to appropriately accommodate these new technologies. In order to illustrate the interplay of business and IT strategies (applicable also to ODSs), we outline a framework developed as a part of MIT's *Management in the 1990s* research project (based on [148]).

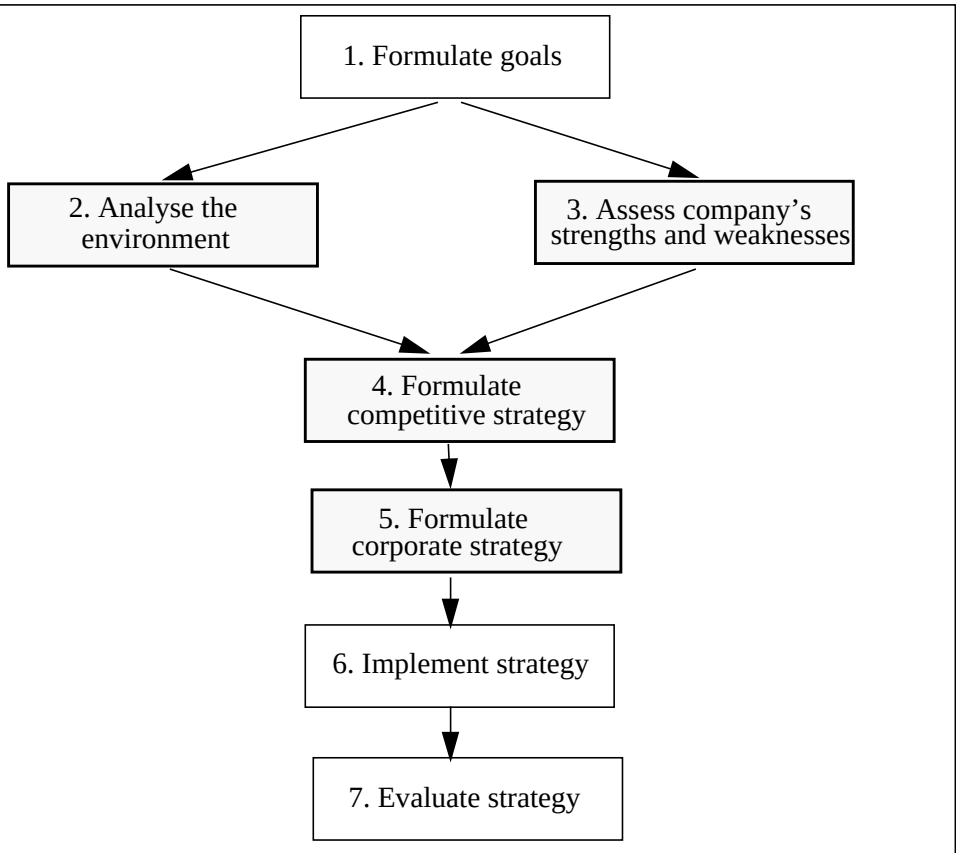

Figure 3.9 The process of strategic planning

This framework is developed along two dimensions: the range of IT's potential benefits and the degree of organisational transformation. In the framework, five levels of IT-enabled business transformation have been proposed, ranging from a pure organisational ef-

ficiency purpose to a complete new philosophy of business scope redefinition, as follows.

1. *Localised Exploitation*. This includes the deployment of isolated and standardised applications and systems in organisations departments, with the aim of responding to some operational problems (e.g. a customer-order entry system, e-mail etc.). In this case changes to the business process are minimal. The use of such an off-the-shelf system cannot be regarded as a competitive advantage, since competitors can deploy the same system. However, if a standard system is customised and a value is added to it, such a system can produce an increased customer satisfaction. For example, the installation of a toll-free number can be motivated by a focus on differentiation and strategic effectiveness, rather than efficiency alone. Therefore, no single application from this category can be regarded as strategic. Rather, it can provide additional benefits, if accompanied with the appropriate business process change.
2. *Internal Integration.* This transformation encompasses two types of integration: technical interconnectivity of different systems and applications through a common (standardised) platform, (e.g. based on the RM-ODP in a case of ODSs), and business process interdependence (of organisational roles and responsibilities across distinct functional lines). The business process interdependence promotes team work and minimises possible lack of communication, so important for having an efficient end-to-end business process. These business aspects should be specified within the ODP enterprise viewpoint of a supporting ODS architecture. A typical example of such an application, in which these issues are of paramount importance is Computer Supported Cooperative Work, as also stressed in section 2.7.

While the previous two levels can be regarded as evolutionary (as they required minimal changes to the business processes), the next three levels can be regarded as revolutionary.

3. *Business Process Redesign*. This level involves a significant changes in business of organisations, as it is based on the view that the new logic of organisations should be predicated on current and emerging IT capabilities. As opposed to current business practices, which are results of the industrial revolution, this level is based on benefits of the information age. As a result, the benefits of IT are not fully realised if only superimposed on the current business processes. Knowing this, a company should first understand the rationale of the current business process. It should then initiate

business process redesign by ascertaining the significant changes in key competitors' business processes (especially those of new entrants, as they can use the benefits of new technologies more efficiently), so that it can formulate appropriate responses beforehand [148]. Business process redesign is particularly beneficial when the scope of this is extended beyond an organisational boundary. The significance of this requirement is recognised by many ODS platform designers, as well as ODP standardisation. This was also driving force for developing our business contract architecture, in chapter 6.

4. *Business Network Redesign*. This level represents the redesign of the nature of exchange among multiple participants in a business network through effective deployment of IT capabilities. It is envisaged that ODSs will provide an important impetus towards such changes. The scope of business network redesign are thus broader than efficient *transaction processing* (e.g electronic data interchange, EDI), based on standard protocols. It may include additional activities, such as:

   a) Inventory movement (e.g in manufacturing, interconnected information systems trigger movements of materials from one stage to another). This should be governed by standard business contracts among the relevant participating businesses. Potential benefits are not only in administrative efficiency, but also in operational efficiency.

   b) Process linkage expands the scope of business network redesign in a fundamental way, e.g. the design stage of one organisation can be linked to the manufacturing stage of another organisation, in a chain through a common CAD/CAM/CIE platform. This type is characterised by specialised contracts or strategic alliances in which each party agrees to the relationship on a mutually beneficial basis. This business transformation represents the core of a new organisational form, frequently referred to as *network* organisations.

   c) Knowledge leverage, which refers to a richer, unstructured information exchange within an intellectual network (as opposed to structured EDI platforms), that crosses physical, organisational and geographical boundaries. The potential benefits are in leveraging critical sources of knowledge and expertise in a broader domain than possible without the functionality the technology offers (e.g. different experts not present in the operating room can solve unexpected complications).

5. *Business Scope Redefinition*. The emerging ODSs blur the difference between organisations and provide an expansion of business scope, providing additional sources of revenues for organisations. Key strategy variable of past, such as economies of scale, mergers and integration leading to vertical integration are giving way to new strategic concepts, such as joint ventures and networked organisations. The capabilities of ODSs thus represent a fundamental source of business scope reconsideration to redefine the ‘rules of the game’.

These five levels are depicted in Fig. 3.10. We note that a variant of this methodology has been used to analyse business processes of a health organisations, as presented in [33].

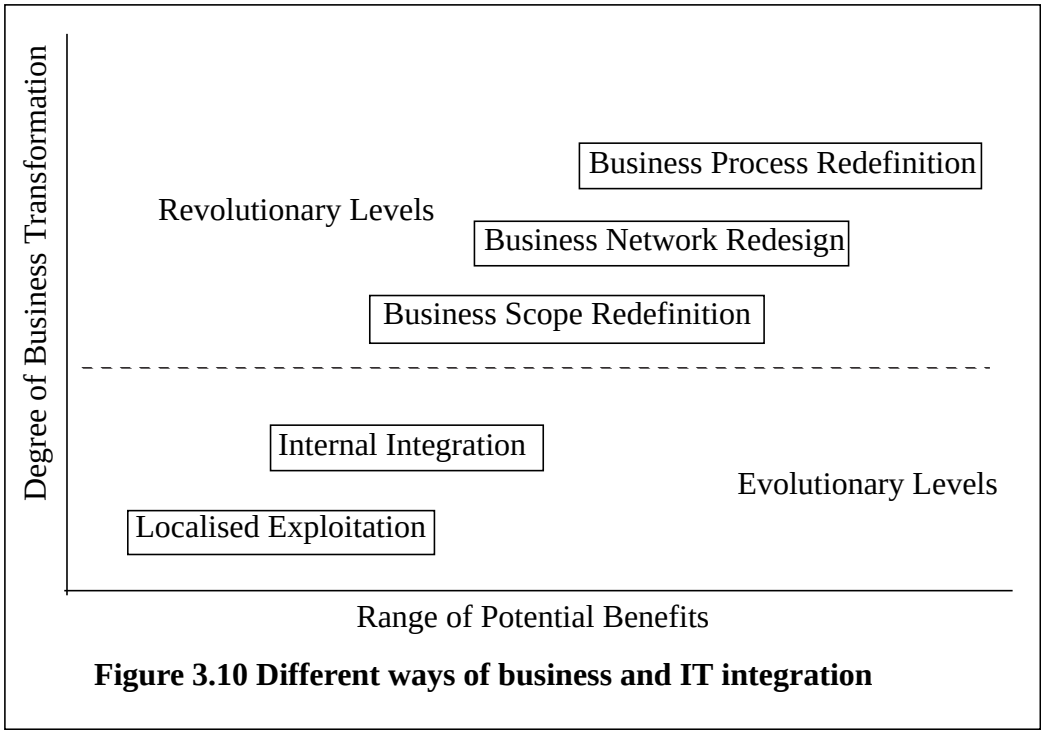

Figure 3.10 Different ways of business and IT integration

3.8 Game theory

Game theory[11], which for more than two decades was a neglected and remote area in mathematics, has grown rapidly and has become a major tool for the construction of new economic models. In fact, the eighties are the years in which economic theory has been revolutionized by game theory [127]. These models provide for a rigorous analysis of sit-

11. This theory was originally developed by J. von.Neuman and Morgenstern, and reported in their seminal work [105].

uations in which small number of economic agents are involved in a conflict of interests where their behaviour is rational, given their expectations about the behaviour of the agents with whom they interact.

3.8.1 Basic concepts

GT includes a broad variety of different class of problems, but the most usual classification is into cooperative and non-cooperative games. A game is said to be cooperative if the players can negotiate binding contracts that allow them to plan joint strategies. On the contrary, in a non-cooperative game, negotiation and binding contracts are not possible.

In other words, GT is a formal analysis of *conflict* and *cooperation* among rational decision makers (e.g. human individuals, corporations, nations, etc.), while pursuing their own objectives[12]. The essential elements of a game are players, actions, information, strategies, pay-off, outcomes and equilibria (Fig. 3.11). The players, actions and outcomes are collectively referred to as the *rules of the game*. The *players* are the individuals who make decisions, while striving to attain their goals, i.e. to maximise their utility by a choice of actions. *Nature* is a player which is used to represent an exogenous uncertainty, by taking random actions at specified points in the game with specified probabilities. An *action* or *move* by a player is a choice he can make.

For the purpose of representing a game in the context of ODSs, an *extensive form* of game seems to be the most appropriate; it can be used to represent complex games in which the order of moves is important and where some actions are not instantly observed by every player - the scenario indeed applicable to ODSs (as will be seen in chapter 5). The extensive form uses a *game tree* representation, whereby nodes of the tree represent points in the game at which some player or Nature takes an action, or the game ends (outcomes of the game). A *branch* is one action in a player's action set at a particular node.

Players' *information* at any particular point in the game is modelled using the concept of

12. The following material is based on [116].

the *information set*. This is defined as the set of different nodes in the game tree that the player knows might be the actual node, but between which he cannot distinguish by direct observation. If the information set has many elements, there are many values that the player cannot rule out; if it has one element, he knows the values precisely.

A player's *strategy* is a rule that tells him which action to choose at each instant of the game, given his information set. A *strategy combination* is an ordered set of one strategy for each of the players in the game. A player's *payoff* is the *utility* he receives after all players and Nature have picked their strategies and the game has been played out. The *outcome* of the game is a set of interesting elements that the modeller picks from the values of actions, pay-offs, and other variables, after the game is played out.

An *equilibriu*m is a strategy combination consisting of a best strategy for each of the players in the game. A *solution (equilibrium) concept* is a rule that defines an equilibrium based on the possible strategy combinations and the payoff functions. Two of the best known solution concepts are *dominant strategy* and *Nash equilibrium*. The strategy is *dominant* if it is a player's strictly best response (i.e. the strategy that yields him the greatest payoff) to any strategies the other players might pick; a *dominant strategy equilibrium* is a strategy combination consisting of each player's dominant strategy. Since very few games have a dominant strategy equilibrium, a more widely accepted equilibrium is Nash equilibrium. The strategy combination is a *Nash equilibrium* if no player has the incentive to deviate from his strategy, given the other players do not deviate.

The *information structure* of a game can be categorised in four different ways, i.e. games with perfect, complete, certain and symmetric information. In a game of *perfect information* each information set is a singleton. Otherwise the game is one of *imperfect information*. In games with perfect information, each player always knows where he is in the game tree, no moves are simultaneous and all players observe Nature's moves; thus, the strongest informational requirements are met in this game. Any game of incomplete or asymmetric information is also a game of imperfect information.

A game of *certainty* has no moves by Nature after any player moves. Otherwise the game

is one of *uncertainty*. Under uncertainty, the model used to evaluate players' uncertain future pay-offs is based on maximising the *expected* values of their utilities. This will be exploited in chapter 5 in the course of modelling the influence of uncertainty in ODSs.

In games with *incomplete information*, a player does not know the precise types of the other players; otherwise the game has complete information. Games with incomplete information are represented by letting Nature move first, choosing the *type* of each player (i.e. the strategy set, payoff function etc.). This first move of Nature is referred to as its choice of *state of the world*.

In a game of *symmetric information*, a player's information set at any node where he chooses an action, or at an end node, contains at least the same elements as the information sets of every other player. Otherwise, the game is of *asymmetric information*; a major subject of *agency theory*. As will be shown in chapter 5, this type of games can be used to address the problem of users' unobservability of service providers actions associated with service delivery in ODSs.

We note that the principal-agent problem is a special case of a dynamic two-person game, since principal and agent cooperatively determine outcome functions and then play the game in a noncooperative way.

GT has proven to be beneficial in offering solutions to a number of technical problems which arise in telecommunication networks and distributed systems, owing to its value as a rigorous mathematical paradigm with predictive power. We illustrate some applications below[13].

3.8.2 Applications to telecommunications

GT concepts have been applied to formulate different resource problems in telecommu-

13. Due to space limitations (and probably with injustice to many authors), it is not possible to provide a comprehensive survey of all applications of GT to networks here - rather some typical examples will be mentioned.

nication networks. Examples are the routing problem in a telephone network [54], [91], flow control in computer networks [58], [129], flow control in multiclass [93] and general networks [58], routing in multiservice networks [42], call admission [92] and congestion control [130] in broadband networks. In addition, GT has been applied to model and solve some higher level issues (e.g. pricing) associated with telecommunication networks. For example, [50] investigates stable coalitions between different national carriers when co-operating in the international network to gain benefits from time zones effects. In [30], the role of pricing policies in multiservice networks is studied; it was found that it is possible to set the prices so that users of every application type are more satisfied with the combined cost and performance of a network with service-class sensitive pricing. We also note a recent use of specially designed auctions as a mechanism to facilitate initial allocation of radio spectrum licences to interested Personal Communication Service providers [36]. The motivation behind this is the fact that auctions can be regarded as an efficient mechanism for an initial allocation of licences to the providers who are best able to use them, with the aim of providing benefits to all: providers, consumers and taxpayers.

3.8.3 Applications to distributed systems

Applications of GT to distributed systems issues are not as numerous. While some technical problems in this area (e.g. processor scheduling [88], load balancing [47], [149], file allocation [47], resource management for concurrent computations [149]) have been solved by applying principles from microeconomics (e.g. price mechanism, auctions etc.), GT has been used to provide a supporting theoretical framework (e.g. to prove Pareto optimality of particular resource allocation schemes [47]). In addition, some references to GT can be found in [59], which investigates oscillatory and even chaotic dynamics of distributed computation in which agents have incomplete knowledge and imperfect information on the state of the system.The paper investigates the analogy between distributed computation and biological ecosystems/human economies and in relation to GT, argues that, in distributed computing, the rationality assumption of GT can be explicitly imposed on their agents, thereby making these systems amenable to game dynamic analyses, suitably adjusted for their intrinsic characteristics.

3.8.4 Applications to ODSs

Relevant results of GT can provide solutions for a range of emerging resource issues in the context of ODSs. It has been mentioned that the problems which are a manifestation of the asymmetric information phenomenon, can be dealt with by using a *non-cooperative* game of asymmetric information (as will be elaborated in chapter 5). A representative of such a game is the *principal-agent* model. On the other hand, results of *cooperative* game theory can be beneficial in many circumstances, e.g. enterprises forming coalitions to exploit economies of scale, scope and probably more importantly, *information*.

We believe that there is a broad scope for future applications of GT results to ODSs. Firstly, one can extend present studies to the case of the game set in a multi-player scenario. This will, for example, be of relevance for Computer Supported Cooperative Work (CSCW) applications. As a candidate, agency theory could provide results from studies of agency problems in a team setting. Secondly, some concepts from cooperative GT can be applicable for modelling general aspects of *federation* in ODSs. A particular test example can be the ODP trader federation concept [11]. This highlights another direction for research, which is to look at the function of trader in different organisations, including emerging network organisations which operations are facilitated by the advances of IT. Some initial thoughts about the use of the ODP trader, from the economic standpoint can be found in section 3.11.

3.9 Full picture

We can now compose a complete picture (Fig. 3.11) which includes all the economic theories mentioned in this chapter [100]. This picture summarizes our identification of those economic disciplines which are relevant to ODSs, as well as related business fields.

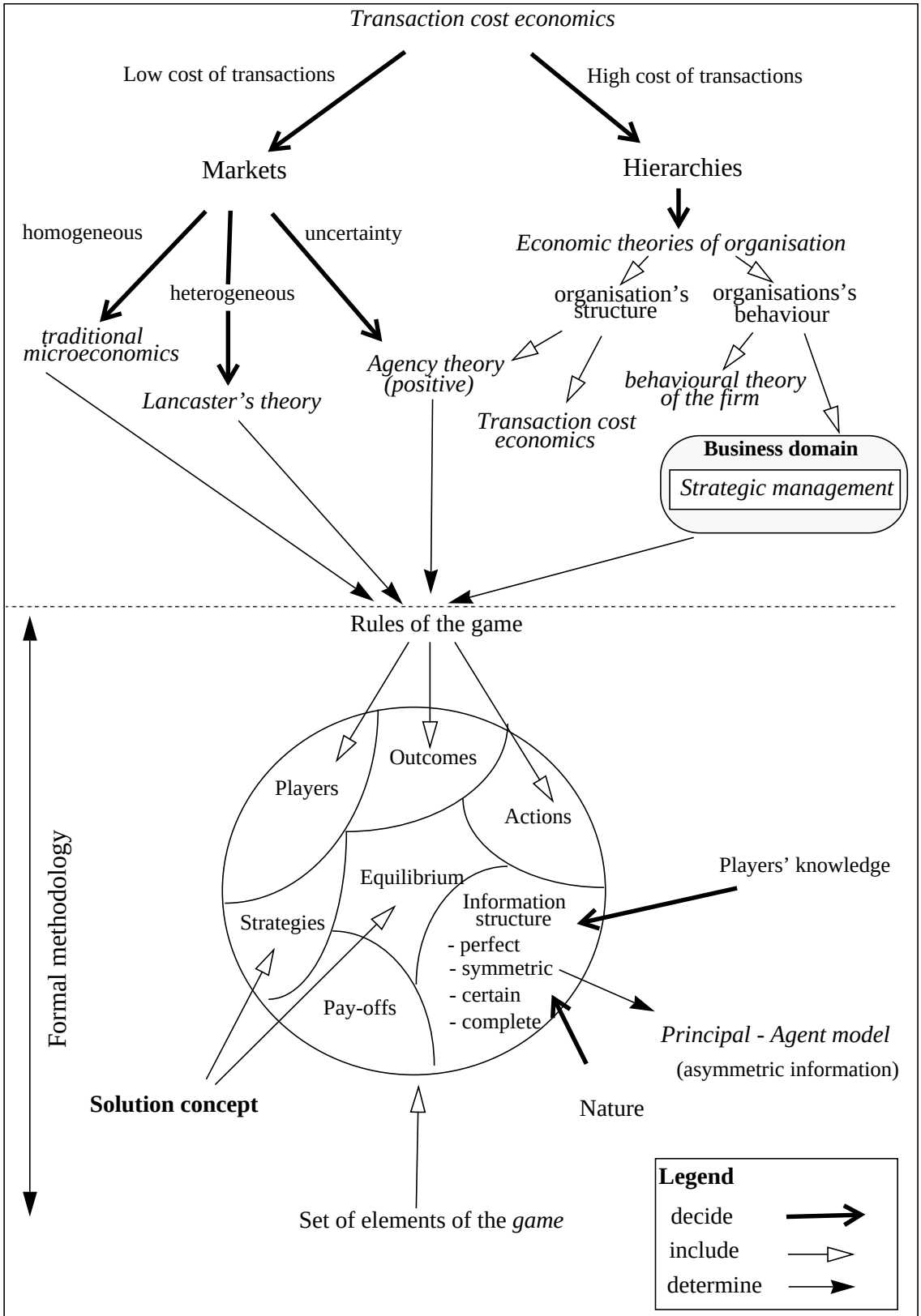

Figure 3.11 Complete picture of economic (business) theories relevant to ODSs

Transaction cost economics can be used to determine whether economic transactions should take place across markets or within organisations: this being determined by the transaction cost minimisation principle [151]. Further, if a market is chosen as a suitable governance structure, then:

- traditional microeconomics can be utilised if the market is one of homogenous commodities and perfect information
- Lancaster's theory of consumer demand can be applicable if the market is one with heterogeneous products or services
- agency theory can be a suitable tool to analyse and model interactions between different economic actors if the market is characterised by uncertainty.

On the other hand, if hierarchy is chosen as the most appropriate governance structure for transactions, one should look at concepts and results of different economic theories of organisation [40], [56], [145]. This includes enquiries into the *structure* of organisations and their *behaviour*. For the explanation of the structure of organisations, the results of transaction cost economics and agency theory are most beneficial. For example, in relation to agency theory it has recently been recognised that the scale and scope of the firm (i.e. its boundary), the capital structure of the firm, various issues regarding separation of ownership and control as well as internal hierarchies of the firm, can all be characterised by different types of agency problems.

In addition to the analysis of the structure of the economic organisations, their behaviour is another important area for an economic analysis of organisations [145]. The behaviour theory of the firm addresses these problems in depth. Further, the behaviour of organisations are also of relevance for a more business oriented type of study, which belongs to the field of strategic management.

The aforementioned theories and disciplines can serve as the basis for determining *rules of a game* (which includes different economic actors and their economic environment) according to which interactions among economic actors take place.These include questions such as, who moves when, who proposes which contract, on which variables does the

game focus etc.

Once the game environment is chosen and the rules of the game are defined, one should be able to start the process of quantitative modelling of interactions between players in a particular setting, and find such solutions which would ensure certain ends, such as Pareto efficiency. If we have asymmetric information, then the principal agent model can be used, as also depicted in Fig. 3.11.

3.10 Impact of ODSs on economic organisations and markets

In the previous sections we have discussed how different economic theories and business fields can be applied to enterprise-related problems in ODSs. However, it is evident from the considerable recent literature on the subject that new IT, in particular ODSs type of information systems, will have a significant impact on the shaping of tomorrow's business operations and on the change of organisations and markets. An understanding of major factors behind this is needed in order to be able to correctly comprehend and predict changes in markets and organisations that are induced by new technologies, so that businesses can successfully compete with other rivals[14]. We discuss these issues from an organisational and market standpoints, but first begin with presenting a suitable framework within which this analysis can be done.

3.10.1 Impact of IT on firms and markets: a framework for an analysis

A comprehensive analysis of interactions between IT and economic activities, can be found in a recent work of Jensen and Meckling [70]. In order to identify major IT factors which impact the structure of organisations (and their interactions at markets), Jensen and Meckling analyse the decision making processes of economic agents. They argue that the opportunity set confronting an individual or firm is a function of the knowledge available. When knowledge is valuable in decision making, there are benefits to collocating *decision*

14. Particularly new entrants, who can possess a better technology and its accompanying organisational structure ('late-mover advantage').

authority with that *knowledge*. This enables the decision maker to make optimal decisions.

There are two ways to bring knowledge and decision rights together: by moving the knowledge to those with decision rights, or by moving decision rights to those with knowledge. The former has been subject of researchers and designers of management information systems, with the aim to transfer the information required for decision making to the decision maker, using the organisation's information system. The latter issue has received little attention in either economics or management [70], but is increasingly becoming the subject of an emerging business concern about organisational redesign and business process redesign, aimed at locating a decision making authority close to the knowledge.

It is important to note that there is a cost associated with moving the knowledge. It involves the use of storage and processing capacity, as well as input and output channels of a relevant resource (e.g. CPU, human brain etc.). This cost of transferring knowledge (information) between agents influences the structure of markets and firms. Since networking, client-server architectures and emerging ODSs significantly reduce the cost of transferring knowledge, they can be seen as a major factor in changing the nature of business operations and transforming the characteristics of organisations and markets, as elaborated in the following.

Impact on organisations

In order to study the synergy between information systems, incentive structures and decision rights in organisations, the structure of an organisation can be analysed via three key variables [25]:

- the *allocation of decision rights* (who is responsible for what actions or decisions)
- the *incentive system*, which defines how decision makers are to be rewarded (or penalised) for their decisions
- a *monitoring* and *measurement* mechanism used to evaluate the actions of decision makers and their outcomes.

Jensen and Meckling explain the structure of organisations as an efficient response to the structure of their information costs (costs of transferring information or knowledge). The assignment and enforcement of decision rights in organisations are a matter of organisational policy and practice. For example, a modern corporation vests all decision rights in the board of directors and the chief executive officer (CEO). Decision rights are partitioned out to individuals and to organisational units by the rules established by top-level management and the board of directors. The CEO enforces the rules by rewarding (and punishing) participants in the organisation. These assignment and enforcement powers are constrained by the laws and regulation of the state and by social custom[15]. However, in order to make a decision, the CEO faces the problem of his own limited capabilities (bounded rationality) to obtain all relevant details directly, and thus normally delegates some authority to other agents, e.g. managers who supervise employees. The employees on the spot generally have a better access to information. Hence, in order for a CEO to be able to make a decision, the information needs to be transferred up the hierarchy, resulting in a variety of *decision information costs* [70]. These include information processing costs and opportunity costs due to delays in communication. These costs are higher as a decision right resides higher in the hierarchy. But this does not mean that decision rights should be located at the bottom of the hierarchy. As explained in section 3.6, the delegation of decision rights brings about agency costs arising from the discrepancy between objectives of agents (e.g. employees) and principals (e.g. managers). An optimal level of decentralisation of decision rights can be found, based on the minimisation of the sum of information and agency costs, as depicted in Fig. 3.12.

Both information and agency costs arise from the cost of acquiring and disseminating information. These costs, in addition to the operational costs determine important attributes of organisations, e.g. allocation of decision rights (as was just explained) and firm size. IT can reduce these costs, by improving the quality and speed of information processing and management's decision making, leading to more centralised management. IT can also provide management with the ability to reduce agency costs through improved monitoring capabilities and performance evaluation, inducing decentralisation of decision making

15. For example, while in many western countries, law defines a 'legal framework' for business transactions, in Japan and some other Asian countries, business transactions are based on the notion of *trust*. This dictates business norms established in dealings between organisations.

[52].

Additionally, IT can influence the size of a firm by changing its cost structure. Since the size of a firm reflects a trade-off between internal coordination costs, operational costs and external coordination costs, and since IT can reduce each of these, the size of firm can be influenced by IT.

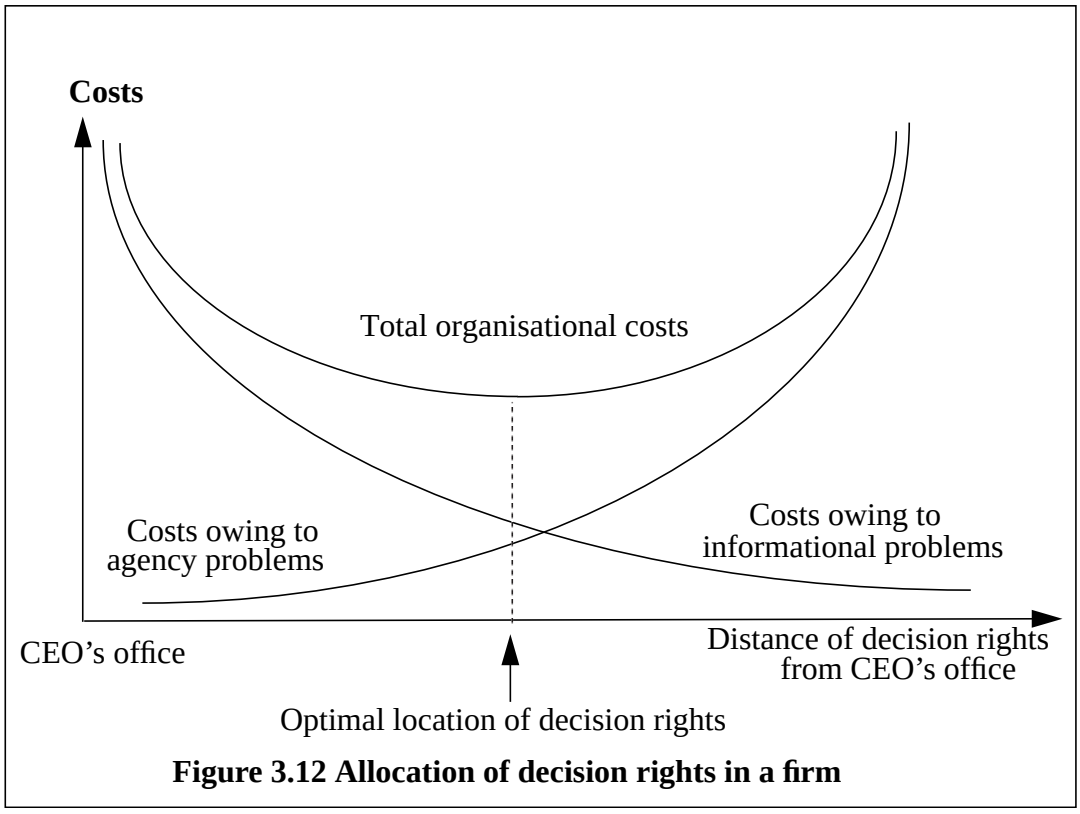

Figure 3.12 Allocation of decision rights in a firm

Finally, the changes made possible owing to ODSs, also bring the possibilities for the formation of new organisational forms, such as *virtual organisations***.** These represent electronically constituted groups, that exist through the space-independent technical artifact such as for example a computer conferencing system [41]. Moreover, these 'organisations' can be formed on a project to project basis; established and terminated according to the project's life cycle.

Impact on markets

From the discussion on transaction cost economics, one can understand the importance of transaction costs as a guiding factor which determines a governing mechanism for transaction execution, i.e. market or a firm. Market transaction costs include the costs of writ-

ing contract and enforcing contracts, but also operational costs such as those associated with search, inventory holding costs and communication costs [87].

IT can reduce operational costs by providing a cost-effective means to access market information and process transactions [52]. IT has also the potential to reduce contract costs, since it can facilitate closer cooperation between firms by information sharing and mutual monitoring. This is a motivation for the development of an architectural framework which can support these functions in the context of ODSs, (as described in Chapter 6).

Impact on strategic planning

Strategic planning has become a major business activity of contemporary organisations. As mentioned in section 3.7, the emerging technologies, especially those that can be put under the umbrella of ODSs, further emphasize the importance of strategic planning. These new technological trends provide an impetus for changes and transformations of organisations in terms of their structure, business process definition and interactions with other organisations. This is evident in the redesign and re-engineering of business processes that many organisations have undergone recently, but also in the changing nature of competitive market forces, which include an increasing globalisation, technological change and technology transfer. As a result of these factors, many firms resort to new strategies such as forming network organisations as described in subsection 3.1.1. An appropriately conceived ODS architecture can significantly facilitate coordination activities within such organisations, at operational, managerial and strategic levels as suggested in [28].

3.10.2 Specific roles of Open Distributed Systems

It is envisaged that ODSs can impact organisation structure and behaviour at markets in the following ways.

- Reduce cost associated with the search for services (e.g. the RM-ODP Trader functionality along with various resource discovery supporting facilities, such as navigation through the WWW browsers [14], [15]).

- Reduce costs of contractual relationships between firms. This includes improved monitoring of actions of parties to the contracts. A supporting business contract architecture, which will be developed in chapter 6, can for example facilitate this.
- Provide more sharing of knowledge of experts, for example via using Computer Supported Cooperative Work.
- Promote establishment of global market for information services.
- Provide an impetus for organisations to include ODS features as an integral part of their business strategies, to increase their competitiveness (as discussed in subsection 3.7.4).

3.11 The RM-ODP trader component: economic perspective

We conclude this chapter by pursuing the example of the RM-ODP trader component, introduced in subsection 2.4.3, to illustrate the concepts we have discussed. As explained there, the trader functionality is an important constituent of an ODS, since it allows users to locate a required service and subsequently establish contractual arrangements with the service provider. The trader component is therefore important not only from a technical, but also from a business point of view, as recognised in the corresponding ODP enterprise specification (subsection 2.5.2). We will extend this specification, with economic driven aspects of the trader. We first discuss possible trader types and different trader owners, followed by the illustration of how each of the economic (or business) disciplines can be applied to this component. We will identify a number of issues that may need to be considered when analysing the operations of the trader from the economic standpoint.

The trader's role is general enough for it to operate in different *organisational settings* (both profit and non-profit) as well as *across markets*, accessible to residential and business customers. The role depends on the specific purpose for which the trader is established.

Table 3.1: The RM-ODP trader operational environments

<table>
<tr><th>Trader type</th><th>Owner</th><th>Short-term goals</th><th>Long-term goals</th></tr>
<tr><td rowspan="2">Universal</td><td rowspan="2">network provider</td><td>increase network use</td><td rowspan="2">a) maximise and maintain network utilisation
b) increase customer base
c) allow flexible adoption of new technologies</td></tr>
<tr><td rowspan="2">maximise trader use subject to current technology base</td></tr>
<tr><td>third party</td><td>the above all indirectly increase trader use</td></tr>
<tr><td rowspan="3">Special</td><td>non-telco enterprise</td><td>various internal management objectives
a) divert use from universal trader
b) more targeted advertising of services
c) more information on special services</td><td>increase business efficiency (and maximise profit)</td></tr>
<tr><td>network provider</td><td rowspan="2">additional goals as for universal trader according to ownership</td><td rowspan="2">may be similar to those for universal trader according to the ownership</td></tr>
<tr><td>third party</td></tr>
</table>

Hence, two broad categories of the trader type can be identified, which roughly correspond to the market and organisational setting. We refer to these as *universal* and *special* trader [99], as depicted in table 3.1. We note that criteria chosen for this classification emanate from the specifics of rules and policies which distinguish markets and organisations. These can for example be: an access type (e.g. restricted, as in case of organisations and non-restricted as in markets), coverage (e.g. global, as in global market, or local as within a company) and service types supported (e.g. general or special service types). However, this distinction may not be so sharp. For example, a specific trader belonging to a particular enterprise, can be made market accessible, if the company finds this beneficial. On the other hand, having a global coverage may not always mean an open access, e.g. a multinational company's trader.

The trader's rules and policies are determined by the traders's *owner*, e.g.:

- an organisation which decides to purchase (a specialised) trader service for its own needs, usually an enterprise which may not have an IT scope
- a public network provider (carrier) on top of which an ODS is implemented (e.g. part of, or the whole national network)

- a third-party service provider.

The economic (business) disciplines identified in this chapter can be applied to address the relevant economic-related problems of the trader, as follows.

1. *Transaction cost economics* (with some *agency theory* considerations) - to determine the best choice for an enterprise, in terms of whether it should purchase trader or use trader service supplied by another party. According to the analysis and conclusions from the previous section, the prevailing factor which decides whether a trader's user will use (hire) some publicly available trader or purchase one, emerges from the spirit of agency theory and transaction cost economics. For example, a user (say an organisation) should calculate the sum of costs associated with the trader purchase (capital investment) and subsequent internal agency costs (if any). These costs should then be compared with the market transaction costs (these include operational costs such as search costs and communication costs, as well as contractual costs, such as costs of writing and enforcing contracts). If the former costs are lower, then the firm should purchase a trader, and thus become the owner of a (specialised) trader. Otherwise, it should use a publicly available (i.e. universal) trader. Once these cost considerations are resolved, and the operational setting is decided, it is possible to define the rules of a particular QoS game, as depicted in Fig. 3.11. We note that there may be some other strategic reasons for internalising trader in spite of out of pocket costs (e.g. a control may be a factor).
2. *Traditional microeconomics* - to analyse what goals are realistic for the trader's owner in short-term and in long-term time horizons (these terms are defined in subsection 3.2.1), taking into account factors which can be incorporated into the trader's production function. For example, the aspects pertinent to the trader's matching operation can be within the scope of trader owners' short-term goals, while the aspects relevant for the trader's selection algorithms (based on dynamic state of servers), can be a subject of the long-term considerations [99]. In general, it may be assumed that in the short-term view, the technology base including the implementation of sophisticated trader features (such as enhanced selection algorithms, reservation of server capacity in advance etc.) and the range of technology-determined services is not variable; that is to say, the owner of a trader has limited means of increasing its outputs in terms of

volume of operations handled and thus revenue generated. In the long term however, these factors may change, together with market acceptance of and increased discrimination with regard to the technology [99].

3. *Strategic management* - to formulate trader's goals (columns 3 and 4 of Table 3.1) taking into account all the relevant factors (as identified in subsection 3.7.2). For example, the primary goal of trader's owner must be to efficiently provide the information about services to users. In relation to the different trader types, owners of the trader in the universal domain will have short-term objective of increasing use of the trader to maximise operating revenues. In the case of third party ownership, this will be the overriding consideration but for network providers a further goal is to encourage greater use of the network from which primary revenues are generated.

   Over the long-term the aim of the network provider will be to maximise and maintain utilisation levels of the network, to gain and retain a high customer base and to evolve readily through the introduction of new technologies. For a third party, the impact of these factors will be indirect, in so far as they increase use of the trader and possibly other services provided by the third party. In terms of both long and short term profit objectives, cost of development, implementation, maintenance and operation need to be considered, while revenues will be derived from server subscriptions and client requests.

   Most special traders on the other hand will tend to be implemented for various internal management objectives which will help in controlling costs of network services, as indicated in the table. In the long term, their goal will be to increase the operating efficiency of the business. Of course third party traders, in particular, will still have largely profit-oriented targets.

4. *Lancaster's theory* - to define the trader's QoS and provide the associated QoS metrics. The achievement of expected and maintained standards of QoS, which are responsive to customer demands, represents an important part of the means of attaining the trader owner's enterprise goals identified above. In subsection 4.5.2, we will provide a detailed analysis of the trader's QoS specification and measurement.

5. *Agency theory* - to quantitatively model the trader service provision in the presence of environmental risk and information asymmetry. This will be extensively studied in

section 5.5, by studying the trader operating in a market environment. The study will be based on relevant results of *game theory,* as a suitable formal model to formulate relevant trader games, in which players are users, providers of information services, and the trader owner.

We note that the economic analysis of different aspects of the trader carried out in this section represents an initial attempt to illustrate some economic concerns. It is expected that once the trader service becomes standardised and implemented[16], some of these issues would be best tested in practice. However, we argue that the understanding of the possible approaches that can be taken in analyses of specific enterprise related aspects of the trader is of high importance for enterprise planners and strategists, as well as IT managers.

16. To the best of our knowledge there are no commercially available implementations of the trader components yet. We note that a service based on the RM-ODP specification is in the process of adoption within the OMG group [136].

CHAPTER 4

Quality of Service Framework for Open Distributed Systems

In this chapter, we present a framework for describing and measuring Quality of Service (QoS) in ODSs. We first highlight the economic and business significance of QoS for different stakeholders and users of ODSs (section 4.1). Following this is a discussion about the inadequacy of traditional (engineering only) approaches, typically adopted in telecommunication monopolies of the past, in treating new QoS requirements (section 4.2). We stress the importance of a *user-driven* approach. It is argued that for such an approach, one first needs to develop a methodology which should serve as a starting point for *conceptualising* the notion of QoS, and further for *measuring* it, on a service-independent basis (section 4.3). In section 4.4 we explain the process of development of such a methodology, based on Lancaster's theory and illustrate this with two examples of ODS services (section 4.5). This initial QoS methodology can be extended with relevant contributions from market research and psychology, to address the problem of determining weights which reflect the importance assigned to particular QoS characteristics (section 4.6). The resulting QoS methodology can be used as a part of different enterprise related modelling in an ODS, e.g. contractual interactions in which QoS plays an important role (as will be demonstrated in

Chapters 5 and 6). Finally, in section 4.7 some comments will be given regarding the developed QoS methodology.

4.1 Introduction

The term QoS, which has been widely used in telecommunications, is increasingly penetrating within the area of computer systems (another indicator that boundaries between previously separate industries are disappearing). Moreover, this term is used in a broader context than in traditional telecommunications: it is not merely an engineering variable supplied by providers, but a concept determined by users. This fact, along with the proliferation of new types of services in ODSs (much more complex than the telephone service), implies the need for a framework which should ensure a common understanding of the notion of QoS among all parties involved. While we accept the fact that QoS can have different meanings for different service types (e.g. for transaction-oriented services, communication-oriented services or some mixture of both, such as networked multimedia services), we argue that a consistent QoS framework is needed which would facilitate conceptualisation and specification of QoS in ODSs.

The need for a consistent QoS framework is augmented by the fact that QoS is gradually becoming one of the key competitive elements in telecommunication and computing industries. Namely, revolutionary changes in telecommunications and advances in computing have contributed to a greater customer awareness and expectations about QoS issues. This coincides with an increased attention to the field of quality management, which has emerged as a result of a requirement (of many industries) for a more accurate and timely response to changing customer needs. It is thus not surprising that several concepts from the field of Total Quality Management are finding their way into the telecommunications and computing industries [112], in order to increase the competitiveness of providers.

Hence, QoS should be regarded as a variable which embodies not only new technological concepts (e.g. multimedia technologies, mobile computing), but equally important, *economic* and *business* concerns. This is evident from the fact that QoS attracts a central interest of four main groups: users, network and service providers and regulatory bodies

[112] (as depicted in Fig. 3.1). *Users* are increasingly becoming aware of the strategic importance which information systems have for their enterprises, and are now more selective when making choices. On the supply side, *network* and *service providers* understand that they need to provide better quality and lower prices in order to respond to market needs. However, for many reasons, a pure competitive environment is not easy achievable. This is particularly true for the telecommunication industry[1]. Due to the significance of this industry for the community, the role of government *regulatory bodies* is to facilitate transition from a previous monopoly to a more efficient competitive setting, with the aim to ensure that benefits of new technologies increase overall social welfare. This is normally done via regulation mechanisms which include the licensing of competition, the authorisation of tariffs and monitoring of QoS [112].

Consequently, a need arises to develop a QoS framework which will facilitate:

- a general and an unambiguous definition of the notion of QoS, on a service-independent basis
- development of a methodology for QoS conceptualisation and specification based on user requirements
- quantification of QoS attributes, i.e. the development of a suitable QoS metric.

In the next section, we present new characteristics of the telecommunication and computing industries, with the aim of explaining why traditional treatment of QoS is not adequate to address the QoS requirements of ODSs.

4.2 Quality of Service (QoS) in telecommunications and computing

The evolving character of telecommunications and computing will be first outlined to emphasize their present competitive character and the role of QoS in this evolution.

1. For example, the distortions of telecommunication markets have emanated from factors such as history, geography or politics. A detailed discussion on this topic is beyond the scope of this thesis. More information on the topic can be found in [112].

4.2.1 Evolving character of telecommunications and computing

The telecommunications and computing industries of the past could be characterised as monopolistic. In telecommunications, national carriers have long had a monopoly in providing telephone service. Their operation has been to some extent regulated by government authorities[2], since telecommunication networks are considered to be an important public resource (similar to other public utilities such as electric and water systems).

On the other hand, and despite being dominated by large multinationals, the computing industry has been more competitive and less regulated than telecommunications since governments have not regarded computing as a vital national resource. As computing technology became cheaper and more mature (at the beginning of this and the end of last decade), other (especially smaller) entrants have started to take up their market share by offering competitive (cheaper and better quality) products. Those that appropriately considered customer needs and their financial capabilities have made significant success.

Similarly, changes in telecommunications, such as technological advances, their lower price (and thus more accessibility to the public) and the use of computing to support service delivery and operations, have provided an impetus for many governments to promote a transition from monopoly to competition within this industry sector. Consequently, a number of new network and service providers which have entered telecommunication market are contributing to more intense competition. Hence, changes in regulatory policies have facilitated an increase in competition and wakening of former monopolists in telecommunications as well.

It is to be expected that these competitive trends will be brought into the area of ODSs, as these represent a convergence of computing and communication technologies. Therefore, as ODSs emerge, they will increasingly represent new competitive environments, which can be termed open information markets. It is worth noting however, that certain aspects of ODSs may be partly regulated by governments, should they perceive that there is a pub-

2. In terms of economics, such a position is referred to as *regulated monopoly* [13].

lic interest which cannot be protected by market forces. For example, government authorities may confine regulation to the underlying communication infrastructure, to some security aspects of ODSs, business contracting policies, or may extend present regulation rules applied to certain services (e.g.banking).

4.2.2 Role of QoS: economic point of view

The regulation in telecommunications has traditionally been centred around tariff issues. This has begun to change in recent times; for example, QoS activities such as monitoring, measurement and reporting have recently become an important subject of regulations (it is now widely recognised, for instance, that cheaper tariffs should not be realised at the expense of a loss of quality).

In addition, QoS has largely been treated as a service provider issue [113]. It has dealt with technical issues, e.g. network performance, and customers have had little direct control over the QoS they purchased. Competitive forces bring changes to this scenario.

While in computing, quality has long been regarded as an important competitive element, it can be said that the first systematic treatments of quality started in the field of Quality Management. This field has recognised the importance of treating quality as a variable defined by customers: a shift from ‘technology push’ of the past to ‘needs pull’ of the future. As telecommunications has evolved, this shift has become apparent in this industry as well [23], and the quality of telecommunications services and products has become increasingly important to end-users, service providers and network manufacturers. For example, this is evidenced in the expansion of the scope and the domain of quality systems [51]. The treatment of quality has started with controlling product quality, and evolved via managing quality of both the product and the associated manufacturing processes, to a scope which also includes the management of the quality of ancillary services such as order entry and billing. This has recently been expanded to include the quality management of processes such as product development, purchasing and distribution, and associated managerial processes of business process quality management, such as quality assessments and strategic quality planning [51].

Therefore, new open environments and competitive forces urge providers to take new approaches to service provision, in which QoS plays one of the central roles. In this new environment, service providers have to take into account the perspective of end users, who experience the QoS on an end-to-end, application specific basis [60]. Service providers must also take into account general *economic* principles, for instance, the fact that there may often be more than one level of quality of a particular service if there is a significant variation in the price/quality combination preferred by different groups of consumers[3]. In other words, QoS should be considered as a highly differentiated commodity. All this suggests that a more comprehensive view of QoS, including its economic aspects is needed.

Another economic related QoS aspect is the fact that QoS is becoming an increasingly important factor of competitiveness in new environments. This is evident from the actions of many interested groups, from all types of players mentioned, as described below.

- *Users* increasingly demand that new systems and services appropriately reflect their business needs. They are also becoming increasingly active in expressing public concern about the QoS through user associations, e.g. the International Telecommunications Users Group, INTUG, and its national members. It is interesting to note that in several of OECD countries, user groups have focused more attention on the QoS issues than on possible rate rebalancing [112].
- *Network* and *service providers* are more aware of new user roles and thus they strive to increase their market share by offering better QoS. To this end they embark on deploying rigorous quality management practices, which promote close and continuous interactions with users. QoS is now regarded as a key strategic factor for providers.
- Independent *regulatory bodies* are becoming more active in their role of addressing QoS issues, and a number of these organisations have commenced related work (e.g ITU-T and ISO). However, these bodies presently place an emphasis on mainly technical indicators at the expense of consumer indicators [112]. In order words, an important aspect of the *economics of quality* seems to have been neglected so far.

3. The optimal level of quality in terms of maximizing welfare is generally provided at the point where the marginal cost of increasing quality equals consumers' willingness to pay for this [6].

From the standpoint of the RM-ODP, economic related aspects of QoS should be specified within the enterprise viewpoint: QoS can be identified as a concept which emanates from an individual agent's enterprise objectives (as these often include particular QoS levels). For example, QoS determines satisfaction of users and competitiveness of network and service providers. However, provision of a service of higher quality implies a higher price to be paid. The notions of cost and tariff structures are important factors in an environment in which there are different service types, different QoS levels and a multitude of user preferences. If there is no notion of cost involved, there is no reason for the user to select anything than maximum level of services. The <QoS, price> combination plays a major business role for agents involved in service provision (as also discussed in subsection 2.5.2).

We have outlined several economic related aspects of QoS so far. Another pressure, which requires an extension of current treatments of QoS comes from new *technologies* deployed within ODSs, which lay the ground for proliferation of new types of services. In terms of the RM-ODP, this means that the enterprise requirements should be related to the corresponding technological aspects. This should be expressed within information, computational and engineering viewpoints. For example, a maximum response time, as a specific QoS variable stated in the enterprise viewpoint, can be related to a different partitioning of its information within the information viewpoint, the appropriate computational algorithms in the computational viewpoint, and replication of data within the engineering viewpoint [97], as also pointed out in subsection 2.4.2.

In the next section, major new QoS requirements which embody characteristics of new technologies will be outlined.

4.3 New QoS requirements in Open Distributed Systems

A number of requirements originating from new technologies deployed within ODSs need to be able to be described as a part of a comprehensive QoS framework. This includes the following.

- *End-to-end QoS* specification. The new QoS framework must be able to specify concepts which include:

  a) the underlying communications system (e.g. ATM transport and switching)

  b) end-systems (e.g. multimedia workstations and optical disk storage)

  c) the distributed application platform (sometimes referred to as ‘middleware’).

- Support for the description of QoS mechanisms specific to distributed multimedia services, such as *orchestration* [60] and *broadcasting*.

- Applicability to *any type* of application: this means that a QoS framework should facilitate specification, implementation and management of QoS, on a service-independent basis. This is particularly important in an ODS, where the number of different types of services (many of which cannot be foreseen in advance) grows rapidly.

- Definition of terms which are *meaningful* to both consumers (users and customers) and producers (network and service providers). This is particularly important, as there are many different views on what is meant by QoS.

- Provision of mechanisms required to support *QoS guarantees*. The new QoS framework must for example specify mechanisms which assure that the contracted QoS levels are maintained.

- Provision of *QoS negotiation* between users and service providers. This can be regarded as a part of a more general, contract negotiation process between parties (addressed in Chapter 6).

- Inclusion of *dynamic* aspects of QoS. In the case of QoS degradation due to unpredictable, external factors, it would be desirable to be able to renegotiate certain QoS parameters, taking into account the new state of the system.

These characteristics need to be appropriately included in a service specification stage; but they also need to be *measured* during (or after) service execution. In order to be able to quantify different levels of service quality, a *QoS metric* is needed. It should provide a means to measure individual QoS characteristics, i.e. different service performance parameters, but also a total QoS, based on these individual parameters. Such a metric is of

particular importance for consumers (e.g end-users), as they pay for the QoS delivered and would need a means for comparison between different service providers to select the best option. The QoS metric can also assist service providers in identifying and appropriately quantifying particular service parameters relevant to consumers, as this may improve customer satisfaction and consequently enhance the provider's competitiveness. Finally, regulatory bodies may also need this metric when prescribing statutory requirements for QoS monitoring and reporting; they can assess the effects of deregulation (in terms of the maximisation of benefits to consumers) and be able to promote efficient operation of markets by improving the information base available to consumers [6].

In the following section we embark on developing a generic QoS methodology which addresses all the above QoS requirements. It is based on an economic theory of consumer demand (i.e. Lancaster's theory) and also includes results from the fields of marketing research and behavioural psychology.

4.4 Service-independent QoS methodology

As highlighted in the previous section, the problem of studying quality issues in general (e.g. within economics, management and related sciences) has gained more attention in recent times, due to needs driven by competitive economic forces. A similar thrust can also be observed in the computing and telecommunication industries, where the following areas relevant to QoS have been researched to date [34], [60]:

1. Communication network topics, in particular:
   a) QoS guarantees in high speed, integrated service networks
   b) resource reservation
   c) QoS configurability
2. End-system concerns:
   a) real-time extensions of operating systems
   b) media scaling and codec translation
3. Multimedia systems:

a) support for continuous media

b) support for synchronisation both to coordinate multimedia presentations and to maintain timing relationships between real-time data transmissions [35].

Our study complements the above areas: we are striving to develop a methodology which would provide guidelines for conceptualisation of the user QoS requirements in a systematic manner. It should give more insight into how one can go about thinking about and describing QoS requirements for services in an ODS, based on user perception. This specification can then be mapped onto underlying resource requirements by using a corresponding QoS architecture, which in turn facilitates the implementation of a supporting information system (as will be discussed in Chapter 6). In addition, this QoS methodology includes a framework which can be used for measuring QoS.

4.4.1 Theoretical basis: Lancaster's theory of consumer demand

In order to develop a general QoS framework which will meet the QoS requirements identified in the previous sections, we have chosen an economic model, based on Lancaster's theory of consumer demand as a starting point. One motivation for this is the fact that QoS itself represents an important economic variable and thus the application of economics appears to be natural. The essential characteristic of this approach is that it facilitates:

- a sufficiently precise definition of the term QoS
- conceptualisation of the notion of QoS, on an application independent basis
- derivation of a methodology for measuring QoS, i.e. a QoS metric.

As stated in section 3.3, according to Lancaster's theory, services can be viewed as bundles of *characteristics* and *quality* refers to variations in *quantities* of characteristics. In fact, it is possible to clearly distinguish[4] (Fig.4.1):

- the (objective) relationship of *services* and their *characteristics*
- the (subjective) relationship of *characteristics* and *people*.

4. For convenience reasons, we include Figure 3.5 from chapter 3 in this section.

Furthermore, the theory suggests that practical studies should be limited to the *relevant* characteristics of a product or service. This means that (as each product potentially possesses a large number of characteristics), the operational use of Lancaster's approach depends on the ability to confine the analysis to a relatively small number of characteristics, with measurable properties [6]. An important further point is that several characteristics may be aggregated into an *aspect* of service quality, thus helping reduce the number of characteristics to be considered.

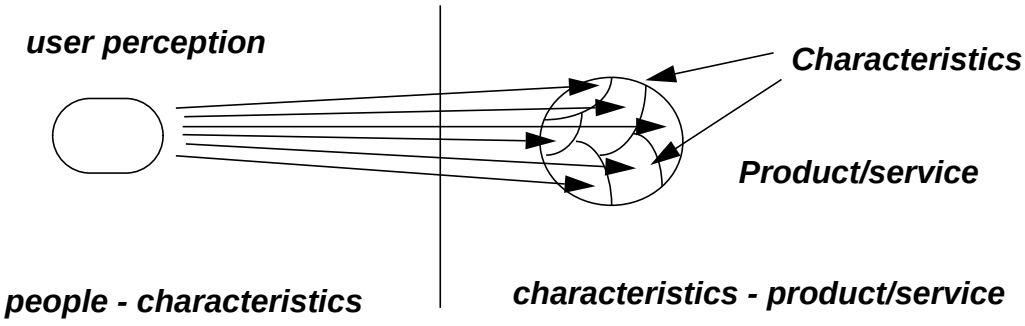

Figure 4.1 Framework for defining quality

4.4.2 QoS definition

It is worth pointing out similarities between this general theory and some of recent work, found in technical documents which deal with QoS issues in the communications and computing arenas. For example, the need for aggregation of characteristics into aspects seems to be also captured by QoS category definition in [68], which is intended to be used to refer to broad grouping of QoS topics. This is also noticeable (to some extent) in a number of definitions of QoS itself. For example, in [65], QoS is defined as 'a set of quality requirements on the collective behaviour of one or more objects'. The ITU-T definition, however, states that QoS is 'the collective effect of service performance which determines the degree of satisfaction of a user of the service' [26]. Capturing user needs in this definition is of a paramount importance and we propose the following working definition:

- QoS represents a set of quality requirements, which are expressed in terms of quantities of service characteristics that directly interact with the utility functions of users.

This is also in line with Lancaster's theory and general QoS definition suggested in [6] and is suitable to be used when, for example, we talk about ODP systems from the enterprise viewpoint and when we take into account the economics of quality. Hence, a service is determined with a set of service characteristics, which are quantitatively measurable variables. An overall QoS is then some function of all service characteristics $QoS = f(Q_{c1}, Q_{c2},..., Q_{cn})$, which is also along the lines of the approach adopted in [13].

4.4.3 Practical framework for QoS specification and measurement

For practical studies Lancaster's theory can be applied by using a decomposition procedure, which consists of the following steps (as suggested in [6]).

1. Identification of relevant user *aspects* of service[5], which reflect major areas of concern for consumers. Appropriate market research techniques can be used for this (as will be discussed below).
2. Identification of *characteristics* that reflect these areas of consumer concern (by aggregating these characteristics together, an *aspect* of a service can be derived). While QoS aspects have predominantly a user-defined (application specific) character, QoS characteristics mainly reflect some technology-related concern (e.g. reliability, response time, availability, accuracy). We note that at a given state of technologies available, it can be possible to define a finite set of QoS characteristics that can be used to encompass all computing and communication technological concerns.

   These two QoS sets, QoS aspects and QoS characteristics, can be represented as rows and columns of the so called *QoS matrix* respectively. An intersection of a row and column (matrix's cell) gives a relevant *QoS parameter.* This matrix can be regarded as a generalisation of the QoS matrix proposed in [123].
3. Provision of operational definitions for performance parameters within the matrix

5. Referred to as *aspects*, hereafter.

cells which allow these characteristics to be measured. The construction of individual QoS parameters should be based on criteria that maximise the effectiveness of the monitoring process, e.g.:

a) coverage of all key aspects of service quality, i.e. those that significantly affect users utility

b) adoption of the parameters which can be measured based on data of acceptable accuracy, and which can be collected on a timely basis and at a reasonable cost

c) consistency of data over time so that the trends in QoS can be determined

d) appropriate protection of commercial confidentiality.

4. Implementation of mechanisms for verifying accuracy of QoS statistics, and for reviewing or enforcing QoS standards. This is of particular relevance for services provided by network providers which are publicly available for general use. Since in such cases market forces cannot always ensure an optimal level of quality (in terms of maximising social welfare), as pointed out in section 4.2, it can be required that these services comply with a number of conformance points, based on the standardised prescriptions of a regulatory authority.

These steps provide a basis for a coherent way of conceptualising the notion of QoS and *specifying* it (first two steps), and a means for *measuring* it (the last two steps). We note that such a matrix is to be derived for each individual service and this will typically rely on appropriate market research techniques. In addition, the rows and columns of the matrix need to be orthogonal. The orthogonality of service characteristics is achieved by the selection of variables that are independent because they correspond to the independent technological parameters. The orthogonality of service aspects needs to be ensured by an appropriate choice of a market research methodology.

To treat QoS using such a matrix is appealing, since the matrix can be:

- used to specify relevant performance parameters for any type of service; it is *service (or application) independent*, though the particular elements of the matrix will vary between services

- applied at various points in the system, where the pair <QoS, price> determines different business arrangements among the agents involved
- exploited by any agent concerned about QoS issues (as identified in subsection 4.2.2)
- used when addressing QoS concerns of different experts involved in strategic planning, specification, design and implementation of an ODS (i.e. applicable within *any* of the *ODP viewpoints*).

From the ODP enterprise viewpoint, the QoS matrix can be seen as a unifying business link between consumers and (network or service) providers. Consumers first state their QoS requirements in terms of the cells' performance parameters. These should then be included as a part of the service/network provider's enterprise specification (which in turn can be used as a starting point for the specification of other ODP viewpoint concerns).

There are several issues which need to be addressed when adopting the above approach. First, it is important to correctly identify a *small group* of service aspects which have a major effect on consumer utility, i.e. to find mechanisms for inferring the value placed by consumers on relevant aspects of service quality [6]. Some methodologies normally used for this are *consumer surveys* (ranking of relative importance of individual aspects), *choice models* (values that consumers implicitly use for alternatives while trading off price with various service aspects), *judgment of experts* etc. The choice of an appropriate method depends on factors such as availability of data and the nature of the service being studied. For example, the consumer survey can be a widely acceptable method from the customers' point of view, but for more complex applications an expert's judgment would be more appropriate due to the possible technical complexity of such applications (e.g. for regulatory purposes).

Second, the proposed framework includes both *objective* and *subjective* types of performance measures (indicators). The objective measures directly relate to the quantifiable characteristics of a particular service aspect and thus measure the associated performance parameter delivered by the service producer (e.g. a number of cutoffs, end-to-end delay)[6].

6. They also embody some subjectiveness, since the identification of key service aspects is based on consumer preferences.

However, subjective indicators can be beneficial in situations where objective indicators are not sufficient to measure all aspects of consumer satisfaction. In other words, the mapping of objective parameters to consumers' satisfaction level can be difficult to ascertain in practice[7]. Therefore, the subjective indicators are used when there is a need to directly measure consumer *satisfaction* with some specific service aspects (e.g. user friendliness, understandability of billing). The optimal balance of subjective and objective indicators may depend on the purpose and audiences for which the indicators are prepared. For instance, in the competitive environment, service producers may emphasise subjective indicators, while regulatory bodies will be more concerned with objective indicators [6].

Finally, we make some observations about the derivation of a *total QoS measure* (or *composite index* [6]), from individual performance parameters. This measure represents a single quantification of service quality, taking into account the different relative importance of QoS characteristics as judged by users (expressed via *weighting* coefficients). This measure can be used in situations in which there is a need for a relatively simple way of identifying trends in the overall level of service quality. However, it is important to note some limitations and approximations of the measure. It is not easy to obtain the precise value for the coefficients, due to the limitation of methods for estimating consumers' priorities. Additionally, these weights will vary between different consumer groups, as a result of their different priorities. Hence, a composite index can be used to provide a useful global measure of service quality, once these limitations are recognised [6]. Different methodologies can be used to obtain these weighting coefficients. This will be discussed in section 4.6.

4.5 Examples of the application of the QoS framework

In order to illustrate the applicability of the proposed QoS framework in an ODS environment we have selected two examples. The first example of a Tourist Information Service (as described in subsection 2.5.2) represents an application domain. The second example is related to an important ODS function, the RM-ODP trading service, as introduced in

7. For example, objective parameters cannot be used to measure the courtesy of telephone operators or the cultural or aesthetic value of television programs [6].

subsection 2.4.3.

4.5.1 A tourist information service

For the purposes of illustration we concentrate on the relationship between an end-user and the Tourist Information Service (TIS) provider and follow the framework developed in the previous section. Accordingly, we first identify the main aspects of the TIS service, and their dependency on technical characteristics, in the following.

1. The TIS aspects of interest for end-users reflect their ability to perform the following major functions.
   - *Searching* through directory information in order to find particular type(s) of tourist information. The quality of the search depends on a number of technical characteristics, e.g response time, ease of use of a search (or browsing) mechanism, flexibility in terms of search options, and so on.
   - *Information presentation* of desired information found after searching. This may involve a multimedia type of information related to a certain tourist or travel component. For example, a video presentation of a particular resort, hotel rooms, surrounding areas, potential entertainment guide, accompanied by high-quality audio information, music etc. The quality of this TIS aspect depends on the characteristics of media types supported, the level of jitter, the accuracy of different streams synchronisation etc.
   - *Booking* of specific accommodation, flight etc. This can for example depend on the number of options provided, reliability of booking (e.g. unreliable 'the last available seat' information), security in terms of how the TIS service provides access to sensitive data of end-users to others.
   - *Billing*, i.e. access to various forms of billing information. The quality of this aspect depends on correctness of bills, e.g. the precision of accounts, travel summary accuracy etc.
   - *Customizing* of the TIS service according to customers' own needs, the quality of which depends on a number of options provided to end-users, e.g. preferred ways of paying and customised booking procedure.

The next step is to identify a set of technical characteristics which determine the quality of the individual aspects identified. Some of these have already been mentioned in the first step.

2. In identifying the characteristics we take into account the specifics of the TIS service and the underlying technological characteristics. Since the major technological concerns are network multimedia services, search (and browsing) services and accounting services, these will determine relevant QoS characteristics on which QoS aspects will depend. The following set of QoS characteristics are identified: *flexibility, simplicity, accuracy, availability, reliability, responsiveness, security* and *coverage*. These QoS aspects and QoS characteristics are grouped to form a QoS matrix given in Table 4.1 (service aspects are depicted as the rows and service characteristics as the columns of the table).

Table 4.1: TIS QoS matrix (end-user side)

	Flexibility	**Simplicity**	**Accuracy**	**Availab.**	**Reliab.**	**Responsiv.**	**Security**	**Coverage**
Searching	X	X	X	X	X	X	X	X
Information presentation	X	X	X	X	X	X		
Booking	X	X	X	X	X	X	X	
Billing	X	X	X			X	X	
Customizing	X	X				X		X

3. For each of the <aspect, characteristic> pairs, the appropriate performance parameter is identified and explained below. These are annotated with 'X' in the matrix cells (otherwise, the relevant matrix cell is left empty). We note that the meaning of the performance parameters may vary, depending of types of users and that these should be determined based on users definitions (and taking into account criteria mentioned in the previous section). The performance parameters given in the table are used for illustration purposes only. The meaning and indicative ranges of values for each of them is given in the following.

a) Search aspect.

This aspect is typically more computational than communication oriented. Hence one would expect that in addition to subjective factors, computing related aspects such as CPU speed, database access time and the quality of the search algorithm would prevail. The related QoS characteristics are listed as follows.

i) Flexibility - this subjective parameter will depend on the type of information searched or a number of different search constraints (e.g. to find the most convenient travel schedule with the cheapest price), and can be expressed for example by a number which belongs to an interval scale (say 1 to 3). For example, in the case of simple searches, such as a hotel search, number 1 can mean a search for all hotels in a specific region, 2 is a better option, e.g. search within a specific region for five star hotels, and 3 is search within a specific region for five star hotels which are ordered based on the distance from a specific venue, e.g. a conference hall. In a case where a search for more complex information is provided, such as for a particular tourist information package, a scale can have different meaning or can be an extension of the above scale.

ii) Simplicity - this is another subjective parameter, described by a scale which reflects ease of use, e.g. 1 is the cheapest option which provides search by means of text, and 3 is the best (and the most expensive) option in which a special graphical user interface (GUI) is provided that highly facilitates the search.

iii) Accuracy - an objective parameter as it can be expressed by a statistic value, e.g. the percentage of correctly found information, or the percentage of information which is up-to-date (level of currency of information).

iv) Availability - can be related to the time interval allocated for the access to the search facility, e.g. a limited number of hours, or 24 hour access.

v) Reliability - this objective parameter can be expressed by a number which reflects a percentage of cutoffs during search. For example, in a case of a local access, this may be 95%, national 85% etc.

vi) Responsiveness - an objective parameter which can be measured by the search delay. This arises due to the communication delay between the end-user and the TIS provider and the TIS database-access time.

vii) Security - probability of a failure to protect the confidential information, say of primary tourist providers.

viii) Coverage - this is another objective parameter that can be measured for instance, by a number of primary tourist service providers (TIS subscribers) which are electronically integrated with the TIS provider.

b) Information presentation aspect.

It is assumed that within this aspect the complexity of QoS management for networked multimedia services will be hidden from a user, but it will still be possible to observe the consequences of the influence of shared networked resources on the presentation aspect. Hence, in a number of the related performance parameters, the multimedia-oriented QoS performance parameters can be used.

i) Flexibility - this measure reflect the choice of media supported to be used for the presentation of the information found during the search. This can be related to some scale interval, e.g. 1 means the textual representation only, 2 can involve black and white images, 3 can include video presentation, 4 is a multimedia option, and 5 is the best option, e.g. a multimedia with high quality resolution and HI-FI audio presentation.

ii) Simplicity - this can mean ease of use (measured by a user determined scale). This can also mean a conformance to standards (e.g. to what extent do all media types conform to some commonly used video types, such as Standard Video, Slow Scan Video, Voice Audio, hi-fi Audio).

iii) Accuracy - this measure is a result of the characteristic of the underlying multimedia system, and includes measures such as jitter, streams synchronisation and rate regulation.

iv) Availability - this objective measure is referred to the proportion of the time that satisfactory network multimedia services are availability and can be expressed in percentages of availability over a finite time interval (e.g. 90% over 30 day interval).

v) Reliability - percentages of cutoffs during the multimedia presentation.

vi) Responsiveness - this is an objective variable measured in time units and includes the set-up delay, transmission delay and delay due to the access to a multimedia information storage.

vii) Security - Not Applicable (NA)

viii) Coverage - NA

c) Booking aspect

i) Flexibility - Number of options offered). For example, 1 may mean no booking option at all, 2 may mean booking of a limited number of travel components, while 3 may mean the option of booking all travel components made available by tourist service providers.

ii) Simplicity - measured by ease of booking, e.g. the use of textual procedure (ranked with number 1), the use of GUI (ranked with number 2), the use of interactive voice system (ranked with number 3).

iii) Accuracy - Level of accuracy of the booked travel component (e.g. various scheduling times, prices, alternatives etc.).

iv) Availability - Time interval in which booking is available, expressed in a similar manner to the availability of search aspect of TIS service.

v) Reliability - this is an objective measures which reflects whether the booked information is really true, expressed in percentages, based on the statistics of measurements, such as the percentage of inconsistent information (e.g. unreliable ‘the last available seat’ information).

vi) Responsiveness - the time (say in seconds) since a booking process is initiated until the response came back.

vii) Security - at a simplest level this can be measured by a scale from 1-3, where 1 can mean no guarantee which would ensure that there would be no leak of a booking information to competitors, 2 may mean that certain booking

information can be guaranteed (e.g. the hotel name), while 3 can mean 100% of the secrecy of booking information.

viii) Coverage - NA

d) Billing aspect

i) Flexibility - this can be a subjective measure which, expressed with a scale from 1-3 represents different billing options, e.g. 1 can mean paying manually, by traditional means, e.g. cheques, credit cards etc., 2 can mean authorising a bank to pay by means of electronic transactions between a TIS provider and the bank, and 3 can be direct electronic payment between the end-user and the TIS provider. This parameter can also be related to a number of various formats for billing reports (e.g. frequency of periodical reports, record logs etc.).

ii) Simplicity - The level of understandability of accounts.

iii) Accuracy - this can represent a measure (expressed in percentages) of the level of correctness of different accounts and other records, e.g. precision of accounts, travel summary data accuracy etc.

iv) Availability - NA

v) Reliability - NA

vi) Responsiveness - is an objective measure which represents the delay between an initiation ad completion of a billing process.

vii) Security - percentage of fraud, whereby the protected information is accessed.

viii) Coverage - NA

e) Customising aspect

i) Flexibility - This can be expressed by a scale which describes different options available to the end-user to customise the service according to her own needs (e.g. 1 can mean only customisation of the preferred way of paying, and 3 can include additional customised features such as the search for a selected set of domains and customised booking procedure).

ii) Simplicity - represents some, user-defined measure of the adaptability of the TIS towards including new features

iii) Accuracy - NA

iv) Availability - NA

v) Reliability - NA

vi) Responsiveness - time required for a customised feature to be implemented.

vii) Security - NA

viii) Coverage - in terms of the choice in specifying a number of different search domains.

4. Once the performance parameters are identified it is necessary to provide mechanisms to verify the accuracy of those parameters. The level of such monitoring depends on the type of application. In case of the TIS service, we envisage that this monitoring can be left to the specifics of commercial relationships between the end-user and the TIS service provider (e.g. a contract between them). However, in a case of dispute related to some QoS issue, arbitration may need to be established. Some of these issues are discussed in the context of business contractual relationships in chapter 6.

4.5.2 The RM-ODP trader

As described in section 2.4, the RM-ODP specification includes the trader component that provides a registration service for service producers (servers) and a selection service to service consumers (clients). We assume that the trader component may also incorporate more sophisticated functionalities. These may include:

- a *selection* of the best service offer for a client from the set of possible matches, based on the dynamic state of servers
- linking up with traders from different administrative domains to form a *federation*, so that a wider market for service producers and a greater choice for service consumers can be achieved.

Based on the description of the trader, the following trading service aspects are identified:

exporting, importing, federating, and billing.

For the trading service characteristics we adopt the same set as in the previous example, namely flexibility, simplicity, accuracy, availability, reliability, responsiveness, security and coverage. These aspects and characteristics are grouped to form the QoS matrix (Table 4.2). Each matrix cell gives a relevant trader performance parameter, the meaning of which is given as follows.

a) Export

   i) Flexibility - In terms of a number of service registration options available. For example, 1 may mean provision of export of static service offers only, 2 may mean inclusion of dynamic service properties (on a change of servers' state) and 3 may mean inclusion of a specific selection algorithm, as discussed in [99].

   ii) Simplicity - Ease of storing service offers. This can depend on the structure of the trader's information tree used to store service offers.

   iii) Accuracy of exported information. For example, this can mean the level of currency of exported information.

   iv) Availability - NA

   v) Reliability - NA

   vi) Responsiveness - delay for registration or update of dynamic service offers.

   vii) Security - percentage of leaked information to other servers or perhaps same as the security aspect of the Import operation.

   viii) Coverage - In terms of number of potential clients that can be supported or a number of service types made available for registration to service providers.

b) Import

   i) Flexibility - Number of options. These could include: provision of a selection facility, choice of searching algorithms/selection criteria, provision of approximate matching/selection, choice of search limits for time/quantity etc. These options can be described using an appropriate scale.

ii) Simplicity - Level of user-friendliness, measured in a similar manner as explained in the TIS search aspect.

iii) Accuracy - percentage of incorrect matches or selections.

iv) Availability - similar to the corresponding parameter explained in the TIS search aspect.

v) Reliability - percentage of cutoffs during trader's import function.

vi) Responsiveness - time for importing. This can be measured via average time over several import requests.

vii) Security - probability of a failure to protect sensitive information.

viii) Coverage - in terms of number of potential servers that can be supported.

Table 4.2: The RM-ODP trader QoS matrix

	Flexibility	**Simplicity**	**Accuracy**	**Availab.**	**Reliab.**	**Responsiv.**	**Security**	**Coverage**
Export	X	X	X			X	X	X
Import	X	X	X	X	X	X	X	X
Federating		X				X	X	X
Billing	X	X	X			X	X	

c) Federation

i) Flexibility - NA

ii) Simplicity - ease of linking traders, ranked according to a pre-defined scale.

iii) Accuracy - NA

iv) Availability - NA

v) Reliability - NA

vi) Responsiveness - time required for the establishment of the federation.

vii) Security - probability of a failure to protect sensitive information.

viii) Coverage - In terms of extent (%) of trader database and service types an exporting trader is willing to share with an importing trader.

d) Billing:

The performance parameters related to this trader service aspects are similar to the corresponding aspect of the TIS service.

Again, we stress the fact that the form of this table and the meaning of individual parameters can vary from one trader service to another.

Observe that for these relatively complex applications, the matrices are fairly ‘full’ (an empty cell means that the corresponding aspect/characteristic pair is not relevant for the application). Also, some of the values in the matrices are directly measurable, while others are more subjective and require various means for mapping onto objective parameters.

It is worth noting that there may be a significant variation in importance of specific service aspects for different groups of consumers. For example, in the case of the TIS service, there may be differences between business and tourist travellers with respect to the value they assign to the number of options for booking, billing, etc. This is expressed by appropriately assigning weight coefficients to particular indicators. The derivation of these coefficients can be done by using different methodologies. In the next section, we show how a methodology from marketing research and a contribution from behavioural psychology can be used to accomplish this.

4.6 QoS attributes valuation: empirical and analytical approaches

In order to better reflect users’ satisfaction with particular service aspects, i.e. to determine the relative importance which users place on certain service characteristics (weighting coefficients), a paradigm from marketing research, *conjoint analysis* can be used. We highlight benefits and possible difficulties with this methodology and consider how a more general theory, *information integration theory* can be used to avoid the strong assumptions adopted by conjoint analysis.

Information services offered in open distributed environments are one of the best exam-

ples which confirm the need for changing trends in customer/producer relations, especially when addressing quality issues. The most important move is to position customers' 'wants and needs' as the driving force for design and engineering processes. As a consequence, the first step is to evaluate customers' preferences regarding new or modified services - the role of market research, consumer research and related disciplines. This, along with the engineers' role in improving quality attributes by adopting a quality assurance paradigm, leads to a more systematic approach in treating QoS. The need for such a treatment is already recognised ([23], [32], [152]), however, there has not yet been a lot of work done on this topic.

One of the very few published references from the communications field, which deals with the applicability of market research in evaluating customers' views on multidimensional and multilevel attributes of services, is presented in [13][8]. Although the paper focuses on the plain old telephone service (POTS), it highlights the problems of subjective customers preferences in circumstances where several quality dimensions are taken into account. It also gives directions about possible empirical and analytical approaches which may be employed when assessing customer subjective issues.

4.6.1 Conjoint analysis

Conjoint analysis is an *experimental* technique developed in the early seventies, and used frequently in contemporary market research. Its aim is to measure the values people place on individual attributes of a multiattribute product or service[9]. The word 'conjoint' refers to the fact that customers values are obtained when the individual attributes of services are *con*sidered *joint*ly (since they might not be measurable, if taken one at a time [29]). Conjoint analysis is regarded as a very attractive experimental tool and can be used to [29]:

- assess customer preferences between competitive alternatives

8. More recent results of this research are reported in [85].
9. In future references we use a more specific term 'service', rather than general expression 'product or service', normally used in this literature.

- identify new (or variations of existing) services which are attractive to customers and thus draw providers' attention to those new features.

There are two basic assumption of conjoint analysis.

- The *additive* nature of people's overall evaluation of services, based on the values of individual characteristics, sometimes called *part-worth utilities*[10]. Experience has shown that this assumption can be valid in many situations (this indeed explains why conjoint analysis has gained widespread acceptance amongst marketing practitioners).
- There are no interactions between attributes.

The conjoint analysis methodology consists of several steps as follows [29].

1. The relevant service attributes should be identified, depending on the nature of the service to be analysed. In doing this, it is desirable that the attributes be important for consumers and actionable so that service providers can control them (this requirement is also incorporated in the QoS framework previously described). The actual attribute levels have to be determined in such a way so as not to overload consumers with redundant (unimportant) information in the data collection phase.
2. Different attribute levels need to be combined in order to provide different joint stimuli to customers. Since the number of combinations may be very large, there may be a need for the selection of a subset of combinations, based on special techniques.[11] Respondents may be asked to describe the utility of a service in two ways: based on *all* attributes (full-profile approach), or to rank combinations of levels of *two* attributes from the most preferred to the least preferred (trade-off approach).
3. This step deals with the selection of the appropriate form of presentation of stimuli (e.g verbal description, pictorial representation), followed by the aggregation of the individuals' judgment (e.g clustering of groups based on various criteria).
4. The analysis of collected data by using the available techniques.

10. Note that in conjoint analysis price can also be regarded as an attribute. In this thesis we treat price as a separate economic variable, as is common practice in economic analyses.
11. For example, the use of the so called orthogonal design technique [29].

One technique used for calculation of part-worth utilities consists of an iterative procedure, as follows [29]:[12]

1. arbitrary utilities are assigned to each level of each attribute of a service
2. total utilities for each of the profiles are then calculated by adding the individual utility values for each of the combinations
3. the goodness of fit between the ranking of the alternatives is then calculated, by using these derived utility values, and the ordering of respondents' judgements
4. part utilities are modified in a systematic way until the rankings of services based on these derived utilities closely corresponds to the ranking obtained by respondents.

The output of conjoint analysis is given in terms of a set of derived (part-worth) utilities for each attribute level.

We now illustrate the use of conjoint analysis by means of an example: evaluation of a new coffee maker as presented in [29].

Example

The purpose of the research presented in this example is to find how consumers evaluate different levels of three coffee-maker attributes (which attribute is most preferred):

- capacity (4, 8 and 10 cups)
- price ($18, $22 and $28)
- brewing time (3, 6, 9 and 12) minutes

The experiment starts by asking respondents to rank the coffee maker, in terms of different levels of these attributes, whereby ranks range from 1 (least desirable) to 36 (most desirable). Let the result of the survey be depicted in the table 4.3, as follows.

12. Another approach is based on the utilisation of regression techniques.

Table 4.3: Responents rankings of a product

Capacity	4 cup			8 cup			10 cup		
Price	$18	$22	$28	$18	$22	$28	$18	$22	$28
Brewing Time									
3 minutes	17	15	6	30	26	24	36	34	28
6 minutes	16	12	5	29	25	22	35	33	27
9 minutes	9	8	3	21	20	8	32	31	23
12 minutes	4	2	1	14	13	7	19	18	11

Now, based on this table, one needs to find the values customers place on particular levels (part-worth utilities). To this end, the following method can be used.

- assign arbitrary values to part-worth utilities
- modify this initial solution through a series of iterations to improve ‘goodness-of-fit’.

For example, let’s start with the following arbitrary values (Table 4.4):

Table 4.4: Initial arbitrary values assigned

Capacity		Price		Brewing Time	
4 cup	0.2	$18	0.6	3 minutes	0.5
8 cup	0.3	$22	0.3	6 minutes	0.3
10 cup	0.5	$28	0.1	9 minutes	0.1
				12 minutes	0.1

Assuming an additive integration function, e.g. 4-cup, 12 min. brewing time, $28 pot has a utility = 0.4 (0.2+0.1+0.1), one can derive a new table (Table 4.5).

Now, one needs to check if the values from this table correspond to the original customers valuation (of Table 4.3). This, the so called ‘goodness-of-fit’ test, can consist of plotting these the corresponding values from the two tables against the other (Fig.4.2).

Table 4.5: Derived utility values

Capacity	4 cup			8 cup			10 cup		
Price	$18	$22	$28	$18	$22	$28	$18	$22	$28
Brewing Time									
3 minutes	1.3	1.0	0.8	1.4	1.1	0.9	1.6	1.3	1.1
6 minutes	1.1	0.3	0.6	1.2	0.9	0.7	1.4	1.1	0.9
9 minutes	0.9	0.6	0.4	1.0	0.7	0.5	1.2	0.9	0.7
12 minutes	0.9	0.6	0.4	1.0	0.7	0.5	1.2	0.9	0.7

If a satisfactory goodness-of-fit measure (e.g. a monotonicity of the points in Fig. 4.2) is not obtained in this first step the iterative procedure should be continued. This consists of changing the values assigned to the attributes in a systematic way to improve the fit, until a satisfactory measure between the two orderings is achieved.

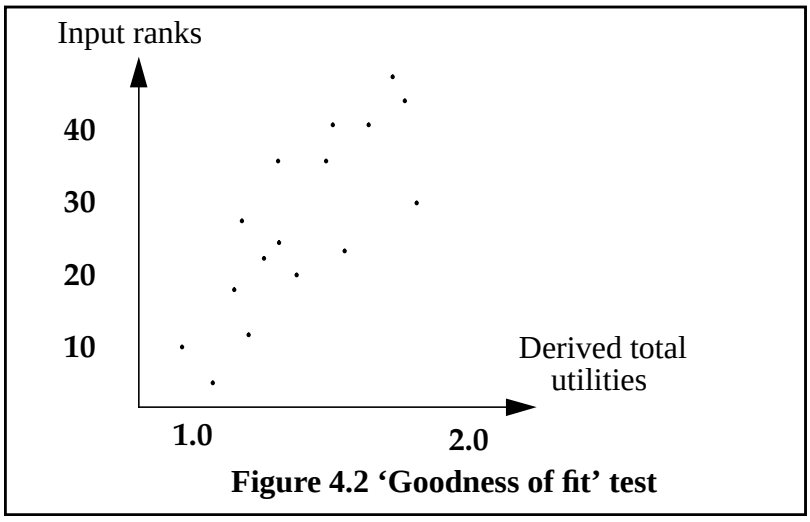

Figure 4.2 'Goodness of fit' test

The utility values from the final iteration represent the solution and the output of conjoint methodology. These utilities are depicted in Fig.4.3, from which it can be concluded that:

- a user's preference for price and size are monotonic
- capacity is the most important and price is the least important attribute (according to the slope of the lines in the figure).

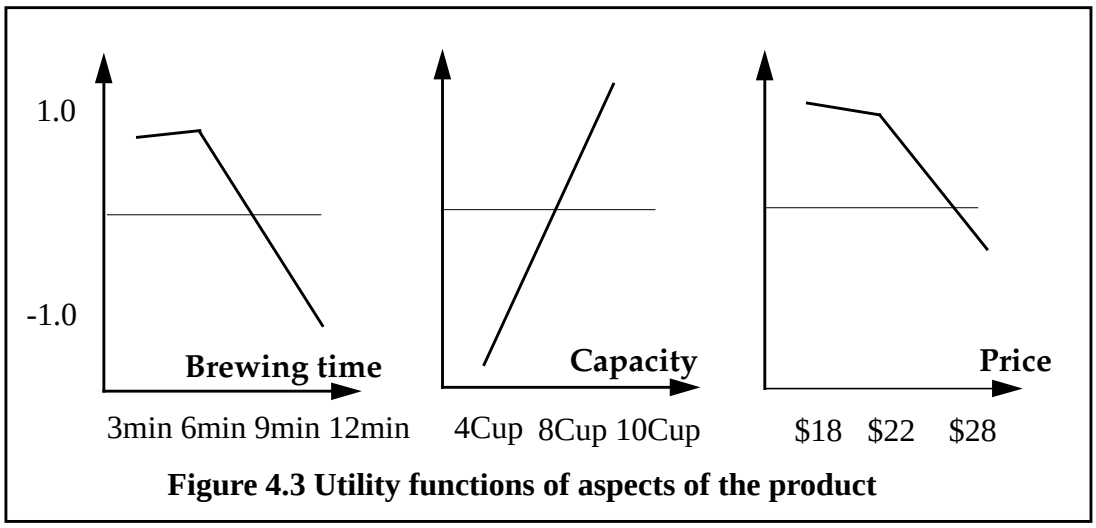

Figure 4.3 Utility functions of aspects of the product

The reliability of conjoint analysis has found to be high. As a result, it has gained widespread acceptance in applied market research. It should be considered as a serious candidate to be used in evaluating the QoS of a multi-attribute character of services in information markets.

4.6.2 Information integration theory

While conjoint analysis represents a very useful experimental tool, it has some limitations, which arise from the assumption of the *additive* integration functional form. For cases in which this assumption needs to be verified, and where other functional forms may need to be considered, another paradigm provides a more general framework for an analysis of customer multiattribute alternatives. This is the *information integration theory* (IIT), which was developed in psychological research of the mid sixties [1]. It can be used to determine the model which most appropriately represents the psychological processes by which consumers combine different service attributes.

The IIT describes individual overt response to external stimuli through a three stage process[13] [1] (as depicted in Figure 4.4).

13. Uppercase letters denote observable (objective) variables, while lowercase letters denote unobservable (subjective) variables.

- *Valuation,* which encompasses processes that map physical stimuli S_i (observable variables) onto subjective counterparts s_i.
- *Integration,* which explains how an individual combines multiple psychological stimuli s_i into a subjective, psychological response r. This represents the central concept of the IIT.
- *Response,* which describes the mapping of response *r* onto observable response *R*.

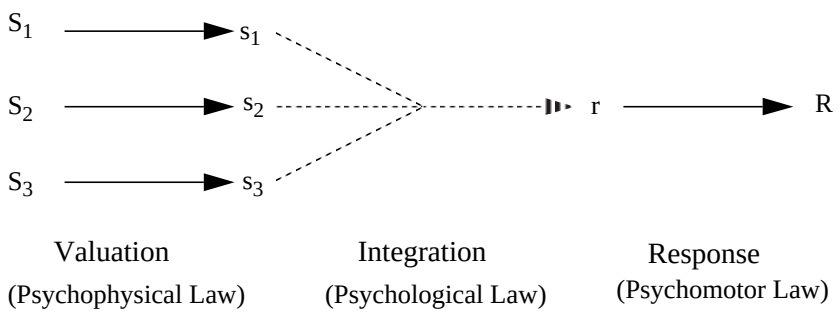

Figure 4.4 Information integration diagram

The IIT has its foundations in experimental work which has shown that the human organism often *integrate*s stimuli based on simple *algebraic* rules, such as adding, averaging, subtracting or multiplying [84]. Consequently, the term *cognitive algebra* was derived, to emphasize this fact. As a prerequisite for applying algebraic rules, measurements of psychological values are needed. The IIT includes a conceptual foundation and practical techniques for these measurements, known as the *functional measurement* paradigm. This includes measuring the psychological value of stimuli, measuring the psychological value of the response and determining the psychological law of integration [84]. Functional measurements methodology is characterised by:

- the assumption of a linear response function R = ar + b
- the evaluation of the functional model, e.g. by using common statistical techniques, such as the simple ANOVA (analysis of variance) techniques [29], if the linearity is proven
- graphic tests which augment statistical tests of the hypothesis about the integration form

- simple techniques for scaling the independent variables if the valuation procedures have confirmed the assumed integration model.

The IIT provides a powerful framework for evaluating an assumed composition rule and validating measures for independent and dependent variables within the model simultaneously [84]. This, along with its compatibility with conjoint analysis makes the IIT very attractive for applied marketing measurements, which need to be employed more rigorously and systematically in information markets.

4.6.3 Information integration theory and conjoint analysis: comparisons

These two methods are similar in their conceptualisation of judgement processes. The major difference is in the fact that conjoint analysis places the emphasis on the valuation stage only, i.e. measuring part utilities of the attribute levels. Also, conjoint analysis assumes the additive algebraic rule, while the IIT does not have this *a priori* assumption. Rather it assumes a linear response function and any algebraic type of integration function, followed by systematic proof of either of these. Hence, the IIT is more general since it focuses on all three stages. In fact, one may say that the IIT involves the use of conjoint analysis, coupled with a rigorous method for testing the hypothesis about the composition rule [84]. The ITT also provides an analytical (algebraic) treatment of total QoS measure.

4.7 Comments

The QoS framework developed in this chapter meets the three requirements identified in section 3.1. It provides the following.

1. General definition of QoS (as stated in subsection 4.4.2). The definition is general, because it is not technology driven. Rather, QoS is defined as a concept determined by users, which reflects the value they obtain from consuming the service. Such an approach provide a service-independent QoS definition.
2. A methodology for QoS conceptualisation based on user requirements. The first step

of this methodology consists of asking users to identify those aspects of a service in question which are important for them. These can then be mapped onto technological characteristics. The combination of aspects and characteristics (i.e. QoS matrix) represents a convergence between users concerns and technological concerns. This also can provide a basis for a QoS specification using formal specification methodologies (this however is beyond the scope of this thesis).

3. Quantification of service parameters. This is done by assigning values to each of the relevant cells of the QoS matrix. Such a matrix can be:

   a) derived by service providers in response to a need to better meet user requirements

   b) used by users (or regulatory bodies) to facilitate monitoring of a service provider or to compare offerings of different service providers

   c) included as a part of a service contract between a user and a service provider (as will be shown in chapter 6).

   In order to reflect the fact that users place different importance on different service characteristics, our framework introduces the concept of weights and proposes two methodologies to derive these.

The value of the QoS methodology derived in this chapter has been illustrated through two examples. However, the nature of this QoS methodology is such that it requires a number of empirical tests. In particular, this includes applying an appropriate market research techniques to: *i)* identify user-relevant service aspects for a particular service and *ii)* calculate weights associated with the corresponding performance parameters (i.e. apply the conjoint analysis technique). We have planned to undertake relevant tests in the context of a local (Government) organisation. Unfortunately, due to several practical difficulties associated with the lack of the resources within that organisation and due to the lack of customer data on which this could be applied, this was not feasible within the time frame of our study.

CHAPTER 5

Designing Service Contracts in Open Distributed Systems

The theme of this chapter is *uncertainty* issues related to QoS delivery in ODSs. More specifically, we investigate the applicability of economic agency theory for the design of *service contracts* which can be used to reduce the negative effects of uncertainty.

Following a brief introduction, we outline new sources of uncertainty regarding services in ODSs and discuss the impact of uncertainty on the resource allocation in an ODS (section 5.2). In section 5.3, we suggest how these issues can be approached through an application of agency theory. Section 5.4 focuses on how one should go about describing agency relationships associated with the delivery of QoS in ODSs: what we call a *QoS game*. Different solutions for this game are also given, followed by a demonstration of the application of agency theory with the RM-ODP trading service, in section 5.5. Several comments about the complexity inherent to agency theory problems and potential difficulties of applying this theory in the context of ODSs will be given in section 5.6

5.1 Introduction

Some new classes of uncertainty which may arise in ODSs have been highlighted in section

3.1. It has been indicated that the uncertainty of ODSs is typically related to *i)* the semantic problems associated with the conceptualisation of new services, *ii)* the unpredictability of the ODS environment in which users interact (in both an economic and technological sense), and *iii)* uncertainty connected with the behavioural patterns of the interacting parties. In the previous chapter we have developed a service-independent methodology for the specification and measuring of QoS to address the potential complexity of services in an ODS and the proliferation of new services. This methodology, in its own right, can be viewed as a mechanism to reduce uncertainty in ODSs. Firstly, it can alleviate some of the semantic problems related to the description of QoS since it involves cooperation between a user and a service provider (SP) in describing specific QoS aspects. Secondly, it can be used to facilitate measurement of the received QoS. At this point we note that the received QoS can be regarded as an outcome which is affected but not completely determined[1] by the service provider's actions, and that by having a means to measure it, one can provide a basis for the outcome-based (not action based) payment scheme. This is particularly useful in an ODS environment in which the actions of SPs are hard or expensive to monitor. In the following sections, these issues will be further clarified.

A QoS methodology like this however, can be regarded as a necessary, but not sufficient condition to fully characterise economic interactions between the parties in an ODS. Assume for example that a user detects a degradation of a certain QoS aspect (by using this QoS methodology say). In many situations, he or she cannot with certainty attribute this degradation to poor performance by a SP or to a problem arising from the ODS environment. This can lead to economic losses to either or both parties. For instance, if a user observes a QoS deterioration he will not be willing to pay the SP for the value agreed previously, but only for the realised QoS value as measured. Even if the SP could have acted as agreed, she would have experienced economic loss as a result of the unpredictability of external factors. On the other hand, a user can experience economic inefficiencies if the SP has private information (not known to the user) and exploits it to her own benefit; a situation quite possible in an ODS technology environment in which SPs are normally experts in their area, while users are typically much less informed within the field. In general, from the economic perspective, uncertainty can be seen as a factor which adds a spe-

1. because of the unpredictability of an ODS environment.

cial dimension to the problem of efficient resource allocation.

All this suggests that in situations in which the influence of uncertainty cannot be neglected, one needs to provide mechanisms which will reduce potential economic inefficiencies that can result. Since this problem has a lot of commonality with economics[2], we adopt the economist's style of thinking in order to address them. We argue that the relevant results of game theory can provide solutions for a range of related resource issues in ODSs.

More specifically, we investigate the applicability of a mechanism which can be used to quantitatively model service delivery in the presence of uncertainty and maximise players' (users' and SPs') utilities under those conditions. This mechanism involves design of *efficient* service contracts between a user and a SP in an ODS and is based on the principal-agent model, an important ingredient of economic agency theory (as introduced in section 3.5). This model is a special case of game theoretic problems: a non-cooperative[3] game of asymmetric information structure. In this chapter, we show how the problem of service delivery can be formulated and solved in an agency setting. In this context, the QoS methodology developed can be regarded as an important prerequisite needed for one party (i.e a user) to measure QoS received as a function of the actions of the others (i.e. SPs).

5.2 Uncertainty of services in Open Distributed Systems

Uncertainty which is associated with services in present information systems emanates mostly from different technical variables such as reliability, availability and responsiveness. However, ODSs bring additional causes of uncertainty, arising because:

- interactions and transactions occur in a more uncertain environment, not only in technical but also *economic* terms (e.g. competition)

2. which can be defined as a science of cooperation with respect to the utilisation of resources [137].
3. Results of cooperative GT can also be utilised, e.g. enterprises forming coalitions to exploit economies of scale, scope and probably more importantly, *information*.

- the interactions between users and service providers center around new variables, such as *QoS*, which are harder to specify, measure and monitor, and which involve some level of subjectiveness.

5.2.1 Sources of uncertainty

Sources of uncertainty in an ODS can have their origins in subjective and objective factors, as depicted in Fig. 5.1. While some subjective factors have been mentioned earlier (i.e. the categories 2 and 3 in section 3.1), objective factors include the following [96].

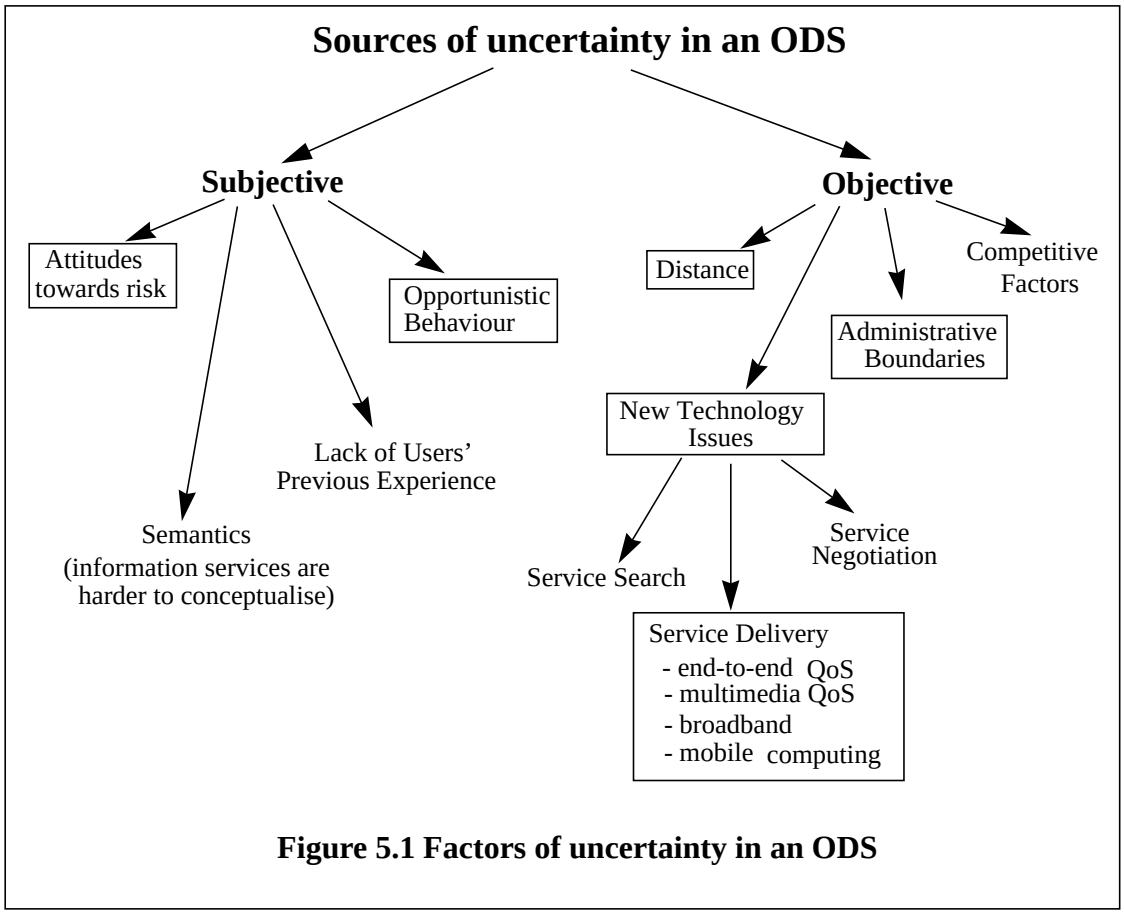

Figure 5.1 Factors of uncertainty in an ODS

1. The unpredictable and dynamic competitive factors of an environment which include:
   a) The number of users and service providers who interact by using underlying resources (this for instance can determine the price of using these resources). This number may be a consequence of the extent to which technologies are adopted by

users, as for example evident in the dramatic increase in the number of Internet users, but also in less than the expected adoption level of EDI technology.

b) Changes in users' QoS expectations and service providers' QoS offerings, which bring a new competitive perspective, QoS competitiveness. This is for example evident in many new telecommunication services, such as Intelligent Network services (e.g. automatic call distribution feature) and personal communication services.

c) New service types that emerge as a result of new technological opportunities and which can increase the demand for existing resources. A recent proliferation of new types of World Wide Web services, such as browsing tools [72], has contributed to the dramatic increase of the Internet traffic.

2. Interactions across large geographical distances (even globally), which bring about unobservability of other parties' behaviour. This, coupled with subjective factors (conflicting goals and opportunistic behaviour), can lead to agency problems (as will be elaborated in section 5.4).

3. Crossing administrative boundaries, since it is harder to get full insight into the status of relevant parameters in external domains, e.g. competence of potential service providers.

4. New technology issues associated with services. These include:
   a) the user's search for, and selection of appropriate service provider(s)
   b) negotiations with providers about service contract terms (e.g. QoS)
   c) delivery of complex services with demanding technology requirements, such as end-to-end QoS guarantees, multimedia QoS aspects (e.g. stream synchronisation), broadband and mobile QoS requirements.

In this chapter we are specifically interested in the uncertainty accompanying service provision in an ODS[4]. This uncertainty arises from the objective factors, e.g. distance, new technologies, crossing of administrative boundaries but also due to subjective factors, e.g. opportunistic behaviour (these factors are captured within the boxes set out in Fig. 5.1). It

4. This assumes that 'pre-service' operations mentioned above, i.e. service search (step a) and service negotiations (step b), have been successfully completed at a previous point in time.

is worthwhile to note that these factors can also be separated into the two groups: the uncertainty of the *environment* and uncertainty arising from the *asymmetric information*.

5.2.2 Types of services particularly affected by uncertainty

From the pace of new services appearing on the markets, it is not hard to predict that there will be a large number of different service types in an ODS. However, some of the services will be affected more and some less by the factors of uncertainty identified above. By adopting the reasoning based on the transaction cost economics [151], in a similar manner as developed in [87], one can identify those services for which uncertainty can be a serious problem. Typically, such services will have the following characteristics.

- Several service aspects of relevance for users. This implies complexity in terms of service description, i.e. the amount of information needed to specify the service aspects which in turn imposes more information requirements on the side of users. Further, it is hard to expect that users can have the same level of information as the service providers who may have developed the service. These are some prerequisites for information asymmetry. Any value-added information service can be a good example, e.g. business insurance policies [87] and a tourist information service described in section 2.5. On the contrary, in the case of simple services, such as an electronic stock exchange (which will be introduced in section 6.4) and electronic trade of commodities, users can obtain perfect information about services (and agency problems do not exist).
- Non-standard form (i.e. idiosyncratic or asset-specific in terms of transaction cost economics). This means that the service is customised for a certain class of customers or for a certain class of application, and that it cannot be readily used by other firms, or in other applications. In the absence of market pressures, providers of such services can engage in opportunistic behaviour, as also discussed in [147].
- Suitable to be exploited by the service providers for gaining a competitive advantage. Any value-added information service, especially one that can play a strategic role for the competitiveness of the service provider can be a possible candidate. Several examples from the eighties confirm this, e.g. early computerised reservation systems

(e.g. American Airlines SABRE system) as well as electronic ordering systems (e.g. American Hospital Supply's ASAP system), which have brought significant profits to their providers. A more recent example of a similar service comes from the insurance sector, whereby an electronic integration service is provided by insurance carers to independent agencies [156]. Another example is an electronic transportation chain management system, increasingly used in port communities [154]. It is interesting how in such a system an opportunistic behaviour can arise, according to the scenario developed in [147]. The authors analyse the trust of a shipping company regarding the performance of a transportation chain management system run by a company in which all shares are owned by a large transportation company. The uncertainty about route evaluation can arise from the fact that the transportation management system can suggest an 'optimal' route to be the one where the transportation company operates, although there are cheaper routes (and that knowledge is not readily available to the shipping company).

It is important to note however, that the above scenarios are typical of the so called 'biased markets', and that regulatory pressures can contribute to change towards 'unbiassed markets' [87], in which the problem of asymmetric information is alleviated. In other words, while at a certain point in time the uncertainty of a particular service can be a problem, regulatory and market pressure can influence a shift towards a more 'perfect' market structure. For example, as an initially non-standard service gains wide acceptance, a requirement for its standardisation at a later point in time can change its level of certainty.

The presence of uncertainty arising from information asymmetry has an impact on the way the benefits of services are spread across parties involved. It is a well known fact in economics that information asymmetry alters the ability of a perfect market to allocate resources efficiently [115]. Bearing this in mind, in the following subsection we will analyse the impact of uncertainty on resource allocation in ODSs.

5.2.3 Impact of uncertainty on resource allocation

The relationship between contract specification and underlying resource requirements in

an ODS is illustrated in Fig. 5.2. The business contract specification requires an understanding of contracts from the economic and legal points of view, as well as typical business practices related to contracting. This includes the identification of important contractual elements, as well as different types of contracts applicable to different situations[5]. The contractual elements specify the *business* level interaction in an ODS environment. A typical commercial contract involves several contractual elements (e.g. payment terms, contract length, obligations and liability of parties etc.). An important element which unifies both user driven and technology concerns is *quality*. It embodies (normally a small number of) broad areas of users' concerns about a service, i.e. *QoS aspects* as discussed in chapter 4. They belong to the domain of *service* level interactions between agents. These in turn need to be mapped to *technology*-related QoS characteristics[6]. This mapping can be expressed by a QoS matrix, the cells of which contain relevant performance parameters, which can be objectively or subjectively measured variables, as developed in sections 4.4. and 4.5.

Therefore, contract specification takes into account underlying resource requirements via QoS specification. Consequently, the level of uncertainty of the provision of QoS, as it is recognised within the contract, has a direct impact on different resource issues, e.g. the way resources are allocated, shared and charged. This uncertainty arises from the fact that QoS monitoring and enforcing is harder to effect and often can involve (unjustifiable) costs. We note here that other contract elements can be monitored with more certainty (as elaborated in section 5.4).

In general, enterprise type interactions between players in ODSs are related to the QoS management in ODSs. This in turn encompasses specification, mapping (between layers), negotiation, resource allocation, admission control, performance maintenance & monitoring, policing and renegotiation paradigms, as developed in [60]. The focus of this chapter is on the mapping of enterprise interactions (embodied in the contract) onto a resource allocation subset of ODS QoS management in order to provide efficient implementation of

5. The classification of different contract types, based on an economic and legal perspective will be presented in section 6.2.
6. It is useful to distinguish between the generic QoS characteristics (e.g. time delay) and derived QoS characteristics, which are derived from the generic characteristics to be applied to a specific/ concrete scenario, such as transit time [68].

services in ODSs. These specific QoS management functions reflect user enterprise objectives and policies which need to be realised in an unpredictable and non-deterministic ODS environment, as indicated previously.

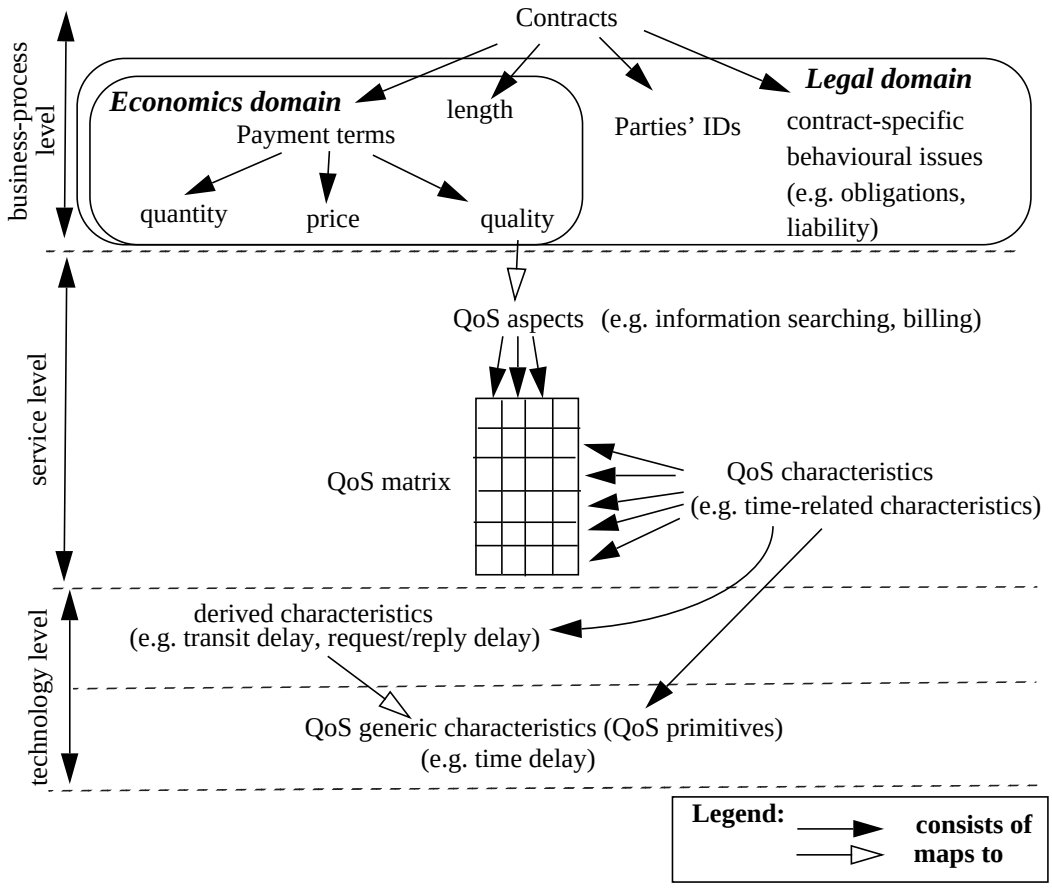

Figure 5.2Relationship between contract specification and QoS issues

We emphasize the fact that new resource problems in ODSs emerge not only as a result of novel technical factors, but also as a result of a number of human-driven, non-technical factors embodied in users' enterprise policies (these are for instance, within the domain of the ODP and TINA enterprise viewpoint). For the commercial success of an ODS architecture, one needs to design resource mechanisms which would take into account the undesired properties associated with interactions between different users of ODSs.

We view topics from information economics such as *optimal incentives* design and a more general optimal *contract* and *mechanism* design[7] [76], (in which the influence of uncertainty on resource allocation is taken into account) as a promising methodology for the

design of such mechanisms in the context of ODSs. As the first step in this direction, we will use a specific theory from information economics, agency theory, to model and solve information asymmetry problems between a user and a service provider in an ODS.

5.3 Application of agency theory to Open Distributed Systems

Agency theory can be seen as a specific paradigm within the broader topics of game theory (GT). In 3.8.1 we have demonstrated how GT has been used to address different resource sharing and allocation problems in telecommunication networks and distributed systems. It is interesting however, that not much work to date has been directed towards incentive compatibility[8] problems (with the exception of the work reported in [30], [129], [130]). Rather, the focus has been on designing mechanisms which achieve some *technical* optimality criteria, mainly at the network and transport layers. Also, most of the research so far has assumed perfect information, e.g. agents are jobs and processors [47]. This was a reasonable assumption for the types of problems dealt with, but unrealistic in the context of ODSs.

As mentioned previously, new characteristics of ODSs contribute towards a number of higher level problems. These emanate from the existence of different parties of different types and with different objectives, and their interaction in achieving these objectives. We contend that the application of GT in this context can be extended towards modelling the roles, actions and interactions of players in such an environment (in a similar way as it is used in GT economic models). GT appears to be the natural candidate to address these novel issues as it can integrate both human and technological issues (via QoS), when modelling interactions between ODS actors while playing what we call a *QoS game*.

To select and formulate the appropriate GT model, the *information structure* of the game should first be identified. From arguments given previously, it is evident that a large class

7. Mechanism design is a broad topic which addresses situations in which there may be multiple agents [76].
8. The concept from economics which characterises those mechanisms for which participants in the process would not find it advantageous to violate the rules of the process (with ultimate goal, such as Pareto-efficient allocation of resources [80]).

of information services in an ODSs can be characterised by asymmetric information among service users and service providers, and hence can be studied in forms of agency relationships (particularly with respect to QoS provision; in the following we refer to these problems as QoS games). Following in the economic spirit, the agency situations can be formulated in descriptive and normative ways. The former will be presented first (in general form), followed by the latter, which is a more formal approach applied to the specific problem in the context of ODSs, i.e. QoS game between users and service providers (Section 5.5).

To summarise the main points from section 3.5, agency theory aims to design the *optimal contract* between principal(s) and agent(s) in the presence of uncertainty. It is now widely recognised that agency relationships dominate many economic activities within organisations and across markets, and we have shown in previous sections of this chapter, why these are also likely to characterise interactions between players utilising ODSs. Further, there are two broad categories of information asymmetry problems: *hidden action* (moral hazard) and *hidden information* (adverse selection) situations.

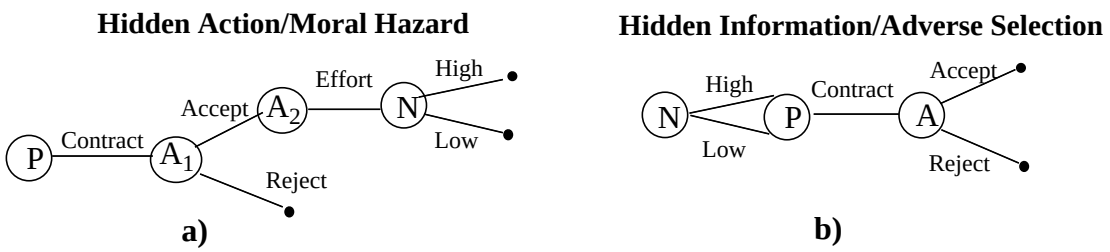

Figure 5.3 Hidden action and hidden information situations

The moral hazard game is depicted in Fig. 5.3a [116]. The principal offers the agent a contract which he accepts or rejects. If the agent accepts it (action A_1), he performs a certain action (A_2), after which Nature (N) adds noise. For example, this occurs when an employer (principal) knows a worker's ability but not his effort. Another example is when a policy holder (agent), can employ different levels of care to avoid a theft, which the insurance company (principal) cannot observe. This is a game of asymmetric, complete and uncertain information (as defined in section 3.8).

In the case of adverse selection, Nature begins the game by choosing the agent's type (his payoff and strategies), unobserved by the principal, after which they agree to a contract (Fig. 5.3b) [116]. An example is a worker (agent) who knows his ability before signing a job contract, but the employer does not; or a potential policy holder who knows her health state before buying an insurance policy. This is a game of asymmetric, incomplete and uncertain information.

The solution to a moral hazard problem is the use of an *incentives* mechanism, i.e. devising a contract so that the agent will in his own interest take actions that the principal would prefer. The solution to adverse selection is so called (market) *signalling*, where the party in possession of superior knowledge signals what he knows through his actions [76]. We now turn to description of agency problems in an ODS setting, and illustrate an AT application with the RM-ODP trading service.

5.4 QoS delivery in ODSs: agency problems and solution

An important characteristic of open information markets, enabled by ODSs is the availability of information services offered at different levels of quality for different prices, providing a greater choice to users. The first step which a user (or service requester, SR) in an ODS would normally perform is to locate an appropriate service provider (SP)[9], based on the required service type and service attributes, e.g. quality, quantity and price. This can be done by using an information broker component within an ODS, such as the RM-ODP trader[10]. The combination of these attributes, i.e. <*quantity, QoS, price*> will constitute a central part of a service contract which binds the SR and the SP. Note that if a service can be regarded as a homogenous commodity (i.e. with a single QoS level), the standard microeconomic supply/demand theory can be applied in order to find an equilibrium (achieved through the price mechanism). However, multiple QoS levels add an ad-

9. Optimal search is another important topic from information economics [76], which is not addressed in this thesis.
10. Strictly speaking this also includes the use of a type management system (to be discussed in section 6.5), but this is not of relevance for the enterprise related issues which we are discussing in this chapter.

ditional complexity to the problem of finding an equilibrium, as will be described in the following.

5.4.1 Motivation for agency theory modelling

Assuming that SRs and SPs are rational decision makers and given a fixed amount of service quantity, the SR will be interested in maximizing her net wealth, i.e. the difference between the monetary value she obtains from the service and the price she pays for it, and similarly, the SP in maximizing his net wealth, which is the difference between the payment received from the SR and costs in providing the service. If *perfect* (or more specifically *symmetric)* information is available to both players[11], an efficient agreement between the SP and the SR can be determined for this simple QoS game. Such agreement is often termed *first-best design of cooperation* [137], and can be arrived at when two players have different <QoS, price> options (and both know these) and successfully complete the QoS negotiation process. The agreement commits the SP to a certain performance in exchange for a certain pay to be made by the SR.

However, the assumption of perfect information is unrealistic in ODSs, as shown in section 5.2. It is likely that one or more sources of uncertainty become non-negligible in the process of service provision, directly impacting players' benefits in such systems. We shall assume that changes in agreed upon quantity and price of a service can be detected (in fact, price of a service can change dynamically, but there are different mechanisms to inform SRs about this, e.g. service contract can contain a clause about this, SRs can inquire about the price through say a trading service). However, a SP's performance regarding QoS element of contract is harder to monitor due to the unobservability of the SP's actions. Hence, given perfect information about price, we see a major source of uncertainty in terms of the QoS delivered to the SR. One therefore needs to consider this uncertainty when designing contracts between players. This is indeed a non-trivial resource allocation problem since it involves:

11. The game theory terms which are used extensively in this section are summarised in section 3.8.

- QoS mapping between a user's QoS view and the corresponding technology variables. As discussed in chapter 4, this is hard problem in itself. To cope with the proliferation of new services and user driven requirements, one needs a generic, service-independent and user-driven methodology for QoS specification and measurement. Such a methodology (e.g. as developed in chapter 4) is a prerequisite for a user to be able to measure QoS as the *output* of the SP efforts.
- Incorporating dynamic effects of the system caused by other users utilising ODS resources. This forms a part of the ODS environment uncertainty. Such effects can be taken into account in a contract design via appropriate distribution function that models this state of Nature.
- Designing an appropriate reward schemes which will *i)* provide incentive mechanisms for the SP performance and *ii)* ensure the optimal spread of risks among the SP and the SR, taking into account their characteristic in terms of their *attitude towards risk*. For example, a *risk aver*se decision maker prefers a certain choice to a risky choice with the same expected utility. A *risk neutral* person is indifferent to these alternatives. A *risk lover* is the opposite to a risk averse individual.

It is important to note that these resource problems are of particular relevance for the specific type of information services, as identified in section 5.2. Since for such services a common denominator of the relationships between users and service providers is agency relationship, we study how AT can be applied to these cases.

Further, in the presence of uncertainty, the first-best design of cooperation is generally not achievable, and one seeks the so called *second best agreement* (see 5.4.2). Following in the spirit of AT, this difference is called *agency cost* and it is borne by the principal, agent or both, depending on their attitudes towards risk. Broadly speaking, agency costs are incurred from discrepancies between the objectives of the principal and those of the agent [52]. We explore these issues in the context of service delivery in ODSs. According to the AT terminology, the SR has the role of a principal and the SP has the role of an agent [97].

5.4.2 Service provider/service requester QoS game: basic model

Firstly, we outline a general *framework* for analysing a game, i.e. a set of recommendations about the GT modelling procedure. The following steps can be used to describe a game [116].

- Identify the *rules of the game*, i.e. *players*, their *actions* and *outcomes.* To this end, one needs to consider where the rules come from for a particular scenario, i.e. to determine the operational settings in which a game is played (e.g an organisation or market).
- Look at the *information structure* of the game and define its type (e.g. a game with perfect, certain, asymmetric or imperfect information). If the game is of asymmetric information, then the models developed within AT literature can be utilised, as will be described below.
- Succinctly define a *formal model* of the game. This includes the order of actions and events, forms of players' utility functions, constraints, and Nature's probability distribution.
- Find an *equilibrium* of the game, e.g. a solution which gives an optimal allocation of resources.

We will follow these recommendations in analysing a QoS game between a SR and a SP in the context of ODSs.

Rules of the game

We assume that the SR has already selected an appropriate SP from possible candidates, so that the influence of other SPs is implicitly incorporated. Namely, competition for agents (and principals) directly influences selection but not QoS delivery. Under this assumption (to which we shall return in section 5.6) there are two players in this QoS game, the SR and the SP. The SR wants to hire the SP, because her wealth (e.g. in terms of her business opportunity) depends on the services which the SP can provide. The SP can offer services in various quantities and qualities. To simplify the model, we assume the case of one unit of service which can be delivered at various levels of quality. Each QoS level involves a particular effort, or disutility to the SP.

Formally, the SR's action is to offer a contract with a pre-defined payment scheme *p* (finding such a payment is the central problem of the principal-agent model, as will be described in the following). The SP's action is manifested by his decision to accept or reject the offer. The SP will accept the offer if it can yield him a utility which is greater than his *reservation utility m*, the minimum that induces him to work. If the SP accepts the offer then he needs to choose an action *x*, from the set of feasible actions *X*, $x \in X$. This action results in a particular level of outcome $y = y(x)$, but also incurs disutility $c(x)$ to the SP.

The order of actions depends on the market power of the players. For example, if many principals compete for one agent this can be modelled by letting the agent move first. In the context of an ODS, this will be the case when there is a small number of particular types of information service providers, in which case their opportunistic behaviour can result.

It is normally assumed that function *y* represents the monetary value and it is increasing in *x*. The SP's utility function U_{SP} is decreasing in effort *x* and increasing in payment *p*. The SR's utility function U_{SR} is increasing in the difference between output and payment.

Information structure of the game

As discussed earlier, for a particular class of ODS services this QoS game can have an *asymmetric* information structure. There are two types of private information which the SP can possess and which the SR may not be able to observe (but would normally be aware of). The first refers to the unobservability of SP's *effort* in delivering agreed levels of QoS (the moral hazard problem). The other type refers to SP's ex-ante knowledge about some variables relevant for QoS which the SR does not know (adverse selection). Hence, there is the possibility of a hidden action or hidden information type of situation, exploited by the SP with respect to service delivery. To illustrate the applicability of AT in modelling the QoS game, we formulate a simple model for the hidden action type of this QoS game.

It is important to first note that agency relationship involves two types of problems. One

problem arises from goal divergence, opportunistic behaviour and unobservability of players' actions. The second problem is one of risk sharing which arises when uncertainty of the players' environment (i.e. exogenous uncertainty) can occur. The former problem will be first discussed in the context of an ODS environment characterised by certain information, via use of an example of a tourist information service (TIS) as introduced in subsection 2.5.2. This will be followed by a more general problem which includes exogenous uncertainty of an ODS environment.

In subsection 4.5.1 we have shown how the quality of the TIS service can be specified and measured. Clearly, the inherent complexity of such a service makes it difficult for a user to monitor the exact performance of the TIS SP. For example, the TIS SP can agree to search all the primary tourist providers in a given region, but due to the vested interests in some of these providers, the TIS SP can contact only this subset of potential providers. Although, a user can request some kind of search logs, this can involve additional costs (and still these logs need be trusted). An alternative solution can be in providing an incentive mechanism to the TIS SP, so that payment depends on some monitoring variable readily available to a user. For example, rather than paying on the basis of the number of search domains, a user can pay on the basis of the comprehensiveness of the search report delivered. Similarly, a primary tourist provider can pay a commission to the TIS SP for each sold service of that primary provider.

Hence, an incentive mechanism of some kind is sought which will induce the TIS SP to perform according to the best interest of its customers (both end-users and primary tourist providers). One payment scheme which provides incentives to the TIS SP can be a linear scheme[12] of the form $p(y) = f + sy$, where f represents a fixed fee, and s $(0 \leq s \leq 1)$ denotes a *share* of output y. If s = 0, the SR pays a fixed fee to the TIS SP, and with s = 1, the TIS SP has an incentive to do his very best. Since we assumed that there was no uncertainty of the underlying ODS environment, the players' attitudes towards risks associated with this uncertainty do not come into perspective. However, in the presence of the exogenous risk of an ODS environment, this changes. Now, the players' attitudes towards

12. This reflects many contractual arrangements observed in the real world, such as the two-part tariff discussed in [71]. There can be a number of other incentive payment schemes, e.g. *threshold* contract [116], which provides incentives in terms of reaching certain target levels of effort.

risk determine the optimal share *s* of such risks. This is a typical principal-agent problem: find an optimal payment scheme which provides *incentives* and also different *risk sharing* between players. For example, due to the unpredictable load of the system between an end-user and the TIS SP, the end-user cannot deduce whether say the low response time of the TIS search aspect is because of the congestion of the system between them, or because of some TIS SP specific cause.

In the following, we will show how such a payment scheme can be derived for a more general case in which there is an environmental uncertainty.

Model of the game: hidden action type

As just indicated, due to the influence of environmental uncertainty of an underlying ODS, the SR cannot observe how well the SP is performing. This state of nature is exogenous to both the SR and the SP (and they are aware of this), and can be modelled as a random variable $\tilde{\theta}$. This uncertainty can result from either numerous technical factors, or due to unpredictable, increased demand from other users, which results in performance/ QoS deterioration for the (SR,SP) pair under consideration. The SR's gross wealth (outcome) $\tilde{y}$ is then a function of the SP's action *x*, but also the exogenous risk $\tilde{\theta}$,

$$\tilde{y} = F(x,\tilde{\theta}) \qquad \text{(Equation 1)}$$

Since the outcome *y* is affected (but not completely determined) by the SP's actions [4], [137], a fee for service can no longer be agreed upon by simple negotiation (as would be in the case of the first-best design of cooperation). As before, to cater for the SP's possible hidden action, the SR is willing to provide incentives for the SP's actions via an appropriate reward that is included in a pay function (which will be paid to the SP after the realization of the output *y*),

$$p = p(y) \qquad \text{(Equation 2)}$$

We assume that the outcome *y* is expressed in terms of monetary income (wealth) which is a transferable and measurable quantity [4]. The net SR's wealth will then be $y - p(y)$.

On the other hand, the SP's net wealth (also after the realization of the output *y*) is $p(y) - c(x)$, where *c(x)* represents the SP's disutility (again in terms of money equivalent). Note that these are the net wealth, after Nature has played out. If we include an environmental uncertainty, the randomness is incorporated by substituting equation (1) for *y*. Thus the following values describe net wealth (i.e. pay-offs in GT terms) of the SP and the SR:

$$\tilde{w}_{SP}(x,p) = p(F(x,\tilde{\theta})) - c(x) \qquad \text{(Equation 3)}$$

and

$$\tilde{w}_{SR}(x,p) = \tilde{y} - p(\tilde{y}) = F(x,\tilde{\theta}) - p(F(x,\tilde{\theta})). \qquad \text{(Equation 4)}$$

Being rational decision makers, the SR and the SP aim to maximize their welfare, derived from their wealth, given by (3) and (4). The welfare can be formalised as the expected values of their utility functions. If $U_{SR}(\tilde{w}_{SR})$ and $U_{SP}(\tilde{w}_{SP})$ represent the SR's and the SP's utility functions respectively, their objectives are to *maximize* expected values, i.e. $E[U_{SR}(\tilde{w}_{SR})]$ and $E[U_{SP}(\tilde{w}_{SP})]$, or their certainty equivalents [137], i.e. $u_{SR}^{-1}(E[U_{SR}(\tilde{w}_{SR})])$ and $u_{SP}^{-1}(E[U_{SP}(\tilde{w}_{SP})])$, where u_{SR}^{-1} and u_{SP}^{-1} represent von Neumann-Morgenstern utility functions [76].

Now, the SR chooses a *payment scheme* $p \in P$, which prescribes incentive for the SP, and 'invites' the SP to accept it. The SP decides whether to accept the SR's offer or not, taking into account his *reservation utility*, represented by a reservation constraint, *m*. Thus, the SP accepts a payment *p*, only if the welfare obtained is not below this constraint level [137], i.e.:

$$E[U_{SP}(\tilde{w}_{SP})] \geq m \qquad \text{(Equation 5)}$$

If the SP decides to accept a reward scheme *p*, he will choose such an action *x* which will maximize his expected value $E[U_{SP}(\tilde{w}_{SP})]$. It is assumed that the SR knows this and will take the SP's decision into account by regarding it as a constraint; given this fact, she will then try to maximise her expected value $E[U_{SR}(\tilde{w}_{SR})]$, i.e. to determine exact values for the coefficients of the optimal payment scheme, given by (2). After the realization of the

outcome *y* is known, the actual payment *p(y)* will be provided to the SP.

Solution concepts

In order to find an *equilibrium* of the game, i.e. to design a *contract* which will be *optimal* for both SP and SR, there are two alternatives. One is more GT oriented, i.e. searches for Nash strategies, which involves a guess that some combination of strategies is an equilibrium, and then testing them [116]. Another analytical approach, which will be used here, is to set up a maximisation problem (set up payoff functions with constraints) and solve it using the first order conditions. This approach can be applied to this game since it is a sequential game, in which the last player's maximisation problem can be embedded in the first player's problem as a constraint [116][13]. Hence, the maximisation problem consists of maximising $E[U_{SR}(\tilde{w}_{SR})]$, subject to the maximal value of $E[U_{SP}(\tilde{w}_{SP})]$ and reservation constraint (5). It is important to emphasize here that the hidden action problem cannot be solved in its general form [137]. Rather, for the equilibrium to be fully described, one needs to assume specific forms of a payment scheme, utility functions of players, Nature's distribution function and the dependence of outcome on risk associated with the environment (environmental uncertainty).

Due to the size of the space of all possible forms of fee functions, it is a common practice in the principal agent literature to assume a simple functional form. The adoption of appropriate simple forms which provide models that are closely related to business interactions in the real world makes the problem solving more practical. This might be justified by the fact that complicated contracts incur transaction and computation costs (e.g. costs of writing and enforcing contracts) [52]. One such frequently used functional form which quite realistically models typical business situations, provides incentives and also different risk sharing balancing is a *linear contract* form [116]:

$$p(y) = f + sy \qquad \text{(Equation 6)}$$

where, *f* is a *fixed* amount, that the SR pays to the SP, and s $(0 \leq s \leq 1)$ denotes a *share* of output *y*; if s = 0, the SR pays a fixed fee to the SP (and thus bears all the risk), and if

13. This however presumes knowledge of the others' risk aversion factors etc. Since this may not be a realistic assumption, an alternative method can obtain this on-line (see section 5.6).

s = 1, the SP bears all the risk. For any other value of *s*, the risk is borne by both players; the balance of which depends on players' attitudes towards risk (as mentioned in section 5.4); the latter in turn determines the form of his/her utility functions. Formally, a risk averse individual has utility function with a diminishing marginal utility, and the utility curve has a positive slope and is concave (e.g. an *exponential* function). A risk neutral person has a *linear* utility function. A risk lover has a convex utility function. The level of risk aversion can be expressed with the coefficient: $\alpha = -u''/u'$, where *u* represents an individual's utility function, and u' and u'' are first and second derivatives respectively. In the standard principal-agent scenario, it is assumed that a principal is risk-neutral (as she can diversify), and an agent is risk-averse (he cannot diversify) [115].

Finally, some comments on the distribution function which models Nature's behaviour need to be made. The form of this function depends on which random variable this function models. In the context of ODSs, this may include an engineering parameter, e.g. reliability, delay etc. But, it is also possible to include other variables such as the unpredictability of other customers' utilisation of the same resources which the SR and the SP need to access (e.g network, trader etc.).

Now, if specific forms for the functions y , $\tilde{\theta}$, $c(x)$, $U_{SR}(w_{SR})$ and $U_{SP}(w_{SP})$ are known, then equations (1)-(6) can be used to determine analytically the coefficients *f* and *s* for *p(y)* and the agent's action *x* for an optimum trade-off between risks and incentives. These variables describe the game's equilibrium, i.e. an optimal contract between the SR and SP (in terms of AT, the second best agreement). Specific forms of these functions will be assumed for the example that follows, the RM-ODP trading service; the form of the optimal solution will also be given and discussed.

5.5 Example: the RM-ODP trading service

The basic concepts of the RM-ODP trading service are given in subsection 2.4.3, while the discussion related to the trader component, from economic perspective is presented in section 3.11. In this section we will extend this discussion with a view on agency problems which can occur in the context of trading.

5.5.1 Agency problems in the context of the RM-ODP trading service

The trader component provides two different categories of services, one to the importers (i.e. SRs) and one to the exporters (i.e. SPs). As a result, two corresponding QoS games can be identified. Looking at the information structure of the trader (TR) QoS games, we will identify information problems related to the quality of TR's service. Several examples of asymmetric information problems in a market setting will be given relating to both TR/SR and TR/SP interactions[14]. For instance, on the TR/SR side, some TR actions, emanating from the owner's policies could be hidden from a SR (ex-post situation), which would have an impact on her welfare, e.g. [96], [97]:

- during the import operation, TR does not search through the agreed upon set of SPs (e.g. 'visits' a smaller number of SPs; this may be the case for both matching and selection); the exact TR's search behaviour will be normally hard for the SR to monitor
- the TR uses random selection, instead of the contracted selection algorithm, which would perform better for the SR
- the TR fails to match and select with promised accuracy (e.g. does not process search constraints or limitations in the agreed upon way).

On the other hand, the TR owner may have vested interests in certain SPs, or form coalitions with them (ex-ante situation). Hence the TR can engage in information hiding from a *SR*, i.e.:

- favouring particular SPs (e.g. hiding the existence of a more acceptable SP from SRs).

On the TR/SP side, the following *ex-post* situations can arise:

- TR's failure to determine *dynamic* service properties of a SP's service as contracted; thus not supplying up to date information about the SP's state to possible SRs
- failure to store the updated SP's offer in the trader database.

14. We use the concepts of trader and trader owner interchangeably in this section.

In addition to these asymmetric information problems, certain parameters related to exogenous uncertainty could also arise (e.g. technological or engineering limitations, increased traffic from other users etc.); hence a whole spectrum of agency problems could occur.

5.5.2 Abstract formulation in GT setting and concrete example

We will illustrate how trader information asymmetry problems can be modelled and solved with an example of an ex-post QoS game between a TR and a SR. More specifically, we study how the TR's induced effort and the coefficients of a selected payment scheme depend on the data and parameters of the model.

We assume the case of a specialised trader [99], which stores a specific set of service types, e.g. service offers of electronic commerce SPs. We also assume a market structure in which there is a competition between similar trader service providers for gaining a wider market coverage towards both SRs and SPs. This has implications on a SR's and TR's attitude towards risk (as discussed previously) and will be incorporated in a part of our assumption set that follows.

Since the trader service is just another service type, we apply the general model developed in section 5.4 to the case of this service; the model is described in terms of relations (1) - (5). However, we still need to select specific forms for: $p(y)$, $U_{SR}(w_{SR})$, $U_{TR}(w_{TR})$, $\tilde{\theta}$, y, *and* $c_{TR}(x)$. Following previous discussion, we assume the following.

- A linear payment function that a SR offers to the TR, such as one given by (6).
- A *risk-neutrality* of the SR; this implies a linear utility function for the SR.
- A *risk-averse* characteristic of the TR, which can be represented by using an exponential function, e.g. [137]:

$$u_{TR}(w_{TR}) = -exp(-\alpha_{TR} w_{TR}) \quad , \quad \alpha_{TR} > 0 \qquad \text{(Equation 7)}$$

where the constant risk aversion factor is:

$$\alpha_{TR} = -u''_{TR} / u'_{TR} \quad \text{(Equation 8)}$$

We note that in the case of utility function (7), the larger this factor, the greater risk aversion. In this specific example we chose a mid-range value for this coefficient., i.e. the TR's constant risk aversion factor $\alpha_{TR} = 0.5$. For a more detailed analysis of the measures of risk aversion in general case, see for example [124].

- The TR component provides different options for the search aspect of the trading service, i.e. quality of search (a more detailed treatment of a quality of trading service is given in section 4.5). For example, the TR can search a different number of trading domains, ranging from a local to a global domain (possibly with a different maximum number of exporters), which requires a different level of 'effort' (measured say, in time units). Formally, we adopt the TR's search action set to be: $x \in X = [0, 1]$, where a larger number from the interval represents a wider search scope, given all other variables are the same. For example, x=1, can mean that the TR agrees to search for a particular kind of the electronic commerce SPs at global level. If x=0, the TR will not engage in search at all.
- The TR will engage in search (i.e. accept the SR's offer) only if it can achieve an expected utility greater than its reservation constraint *m*, expressed by (5). Since the parameter *m* relates to the expected utility, its value depends on how this utility is expressed. For example, this can be a monetary value, e.g. the minimum acceptable income. For illustration purposes, we select $m = 1$. In view of the TR's search operation, this can for example correspond to the minimum number of search domains which will ensure that this constraint is satisfied. However, it could be any other variable, related to the TR's effort.
- The TR's increasing marginal disutility of effort, i.e. the disutility function can be modelled with a quadratic functional form, according to [137]:

$$c_{TR}(x) = x^2 \quad \text{(Equation 9)}$$

- There is an environmental uncertainty, e.g. the other users' utilisation of TR's resources (e.g. CPU time, database access) and communications resources between the TR and the SR, which results in a statistical variation of search response. If the

environment between the TR and the SR were deterministic, then the SR would pay to the TR purely on the basis of TR's search effort x. But in the presence of uncertainty, different satisfaction levels of the SR are not only a consequence of TR's effort, but also this exogenous risk. To simplify the illustration, we assume that the only cause of the environmental uncertainty arises due the unpredictable load caused by different users using the underlying ATM network. In this case, the normal distribution can be a good approximation for the search delay, since this variable is reflected by the user load. Statistically, it can be said that there is a series of random processes (i.e. users) on the path between the SR and TR (quite realistic in the case of such a shared environment). According to the central limit theorem, a composition of such processes tends to a normal distribution in the limit. Therefore, we model this risk (state of nature) $\tilde{\theta}$, by using the normal distribution function with the mean delay $E[\tilde{\theta}] = \mu$, and the variance: $Var[\tilde{\theta}] = \sigma^2$.

- Lastly, the outcome $\tilde{y}$, can be expressed as a linear function of the exogenous risk $\tilde{\theta}$, i.e.:

$$\tilde{y} = x + \tilde{\theta} \qquad \text{(Equation 10)}$$

It is known that when the form of the functions which describe principal-agent model have the so called LEN properties [137] (from **L**inear feasible payment scheme and linear risk function, **E**xponential utility and **N**ormal distribution of risk), one can find the exact solution, based on the fact that the certainty equivalents can be expressed as expected value minus half the variance times risk aversion. Now, the first step is a maximisation of $U_{TR}(\tilde{w}_{TR})$ with respect to effort x. Based on (3), (6), and (9) we have:

$$\tilde{w}_{TR}(x,p) = f + s(x + \tilde{\theta}) - x^2 \qquad \text{(Equation 11)}$$

Owing to the LEN properties [40], the derived welfare is:

$$U_{TR}(x, p) = E[\tilde{w}] - \frac{\alpha}{2} Var[\tilde{w}] = f + xs + \mu s - x^2 - \frac{\alpha}{2} s^2 \sigma^2 \qquad \text{(Equation 12)}$$

Maximisation of (12) with respect to x yields optimal TR's response to the payment scheme (6), given by:

$$x^* = \frac{s}{2} \qquad \text{(Equation 13)}$$

Hence, TR's response to the proposed scheme (6) only depends on the share s.

Now, we need to find which payment scheme will be accepted by the TR, in view of the reservation constraint (5). By using (12) and (13), one can find the value for *f*, so that the reservation constraint (5) is satisfied. This gives:

$$f = m - \frac{s^2 \cdot \left(1 - 2\alpha\sigma^2\right)}{4} - \mu s \qquad \text{(Equation 14)}$$

In order to answer the last question, i.e. which payments scheme (6) in terms of its coefficients *f* and *s*, maximizes the SR's welfare given the TR's response (13) and fixed fee *f*, one needs to set up the SR's maximisation problem. Since the SR is assumed to be risk-neutral, and taking into account (4), (6) and (13) the SR will act to maximise:

$$U_{SR}(x^*, p) = (1 - s)\frac{s}{2} - f \qquad \text{(Equation 15)}$$

After placing (14) into (15), and maximising (14) with respect to share *s*, we obtain:

$$s^* = \frac{1 + 2\mu}{1 + 2\alpha\sigma^2} \qquad \text{(Equation 16)}$$

The optimal fixed part *f* is then obtained from (14) and (16), i.e.:

$$f^* = m - \left(\frac{(1 + 2\mu)^2}{(1 + 2\alpha\sigma^2)^2} \cdot \frac{1 - 2\alpha\sigma^2}{4}\right) - \mu\frac{1 + 2\mu}{1 + 2\alpha\sigma^2} \qquad \text{(Equation 17)}$$

Finally, combining (13) and (16), we have:

$$x^* = \frac{1 + 2\mu}{2\left(1 + 2\alpha\sigma^2\right)} \qquad \text{(Equation 18)}$$

Equations (16), (17) and (18), describe the *equilibrium* of this *QoS game,* and thus the *optimal contract* between the TR and the SR.

Now, if an expected delay between the SR's request and TR's response is $\mu = 1$ time units (say ms), and the variance $\sigma^2 = 4$, the TR will take action x = 0.3, the share will be s = 0.6 and fixed fee, f = 0.95. Hence, using the hidden action model of the QoS game between the TR and the SR, and given the forms of characteristic functions, the optimal contract between the TR and the SR would have the form of the following payment function:

$$p(y) = 0.95 + 0.6y \quad \text{(Equation 19)}$$

and TR's effort,

$$x = 0.3 \quad \text{(Equation 20)}$$

It is interesting to see how a change in the TR's risk aversion impacts the payment scheme. Let us assume a market structure in which the TR SP is more risk averse, with the risk aversion coefficient of $\alpha_{TR} = 1.0$. When inserting this value into (16) and (17) we obtain $s = 0.33$, $f = 0.86$, and $x = 0.165$. Since the TR was assumed to be more risk averse, it is not surprising that the share s is reduced. In addition, since parameters f and s are agreed by both the SR and the TR, the SR expects to get a lower value for QoS ($x = 0.165$) from the TR because the TR agrees only to a smaller value of s. The total amount that the SR is willing to pay in this case is lower than before, leading to a smaller fixed component and a smaller variable component (this need not always be the case: the break-up between the two components may be such that a lower total amount could result from a slightly higher fixed component and a much lower variable component).

This simple example illustrates how one can design optimal contracts between a SP (in this case the trader) and a SR, taking into account asymmetric information and uncertainty. We now identify some potential difficulties of this approach in the context of ODSs.

5.6 Comments

From previous discussions, one can conclude that as ODSs develop they will increasingly resemble complex economic systems in which interactions between parties take on new forms and bring novel requirements. For certain classes of services in which uncertainty

can be a serious problem, as identified above, the appropriately designed contracts can reduce such an uncertainty, in a manner similar to the real world scenarios. In developing an analytical model for the design of efficient contracts, we have adopted several assumptions which deserve the following comments.

We begin with a general comment regarding the QoS game, described in section 5.4. It is obvious that the SR needs to have some way of *measuring* the outcome *y (*which is basically QoS obtained), in order to be able to determine the actual payment which needs to be transferred to the SP. Hence, one requires an appropriate *QoS metric* as a prerequisite. This metric ought to be service-independent, that is to say, applicable to a wide range of services available at information markets. Such a methodology is presented in chapter 4.

As reported in [43], contributions from agency theory are significant in both theoretical and empirical domains. It is however worth noting the following two points in relation to the applications of the principal agent model in the context of ODSs (these also had implications on our model presented in section 5.4).

1. The fully general analysis of the principal agent problem remains to be conducted [71]. In particular, this includes an investigation of *i)* the situations in which there is one principal with multiple agents (this has only recently been given serious attention in the principal agent literature) and *ii)* a scenario where agents of several different principals compete with one another (such game playing agents have been almost entirely neglected in the principal agent literature) [71]. We believe that future contributions of agency theory in these areas would be most beneficial for several related applications to ODSs, in particular those which include the influence of competitive forces on the model of the QoS game. For example, it would be interesting to investigate the influence of the number of SRs (and the number of requests from each of them) on the SP's QoS offerings. In addition, the number of SRs and the nature of interactions in an ODS will influence the SP's attitude to risk: for a sufficiently large number of SRs and in a situation in which each of the SPs can contract with a large number of SRs, one can assume risk neautrality on the part of the SP.
2. Even in relatively simple principal agent settings (e.g. a single principal and a single agent), the mathematical model can be very complex. In relation to this, we note the

following.

a) The solution of the principal-agent model for the optimal sharing rule in a general setting involves technical issues which are both subtle and profound, as discussed in [21]. Analytical solutions are very rare and can be obtained only by imposing restrictive assumptions on the form of utility functions and probability distribution of the outcome. But because of the usefulness of the principal-agent paradigm and the lack of exact analytical solutions, different *numerical* techniques could be employed to determine optimal contracts [21]. It would be interesting to study how these can be used in the context of ODSs.

b) A related problem is reflected in the method for finding the equilibrium strategies presented in section 5.4. This method is based on a strong assumption, since it presumes that the principal knows all relevant characteristics of the agent (i.e. utility function, disutility, reservation level, risk aversion factor). As this is hardly realistic in the context of ODSs, and due to the inherent complexity of the problem, some other solution procedures need to be employed through which information about relevant characteristics can be exchanged between users and a system; thus removing the need to have *a priori* knowledge of these. For example, this can be achieved *on-line*, by employing iterative or hill-climbing methods of solution, in a manner similar to that in [129].

c) A more complete modelling than that presented in section 5.4 would involve a significantly complex mathematic apparatus, which would make the exposition more obscure. We chose to adopt several (perhaps unrealistic) assumptions to simplify exposition at the expense of the precision of the formal modelling[15].

3. Agency theory provides a starting framework for designing contracts in the presence of *i)* the environmental uncertainty of an ODS in which users interact, and *ii)* the users' bounded rationality (as discussed in sections 3.4 and 5.2). This framework can be applied to *human decision makers* or *intelligent agents* who act on their behalf. We anticipate that the intelligent agents will increasingly replace human decision makers and automate the activities currently performed by people. Consequently, when a self-

15. Besides, the aim of our investigation in this chapter is primarily to make IT professionals aware of the problems and possible practical solutions (and not to further study the principal agent problems).

interest of the agents cannot be neglected, this characteristic need to be able to be incorporated in the contracts between such agents.

The following comments are worth rasing in relation to the trader example presented in section 5.5.

1. Nothing has been said about modelling the SP-trader interaction. This is because there is no essential difference between the trader's interaction with the SR and with the SP in as far as the model of games is concerned. In fact, the only differences are in the trader's QoS attributes delivered to either side, which are catered for in the QoS metrics and in different forms of utility functions, probability distribution and disutility of the SP and the SR. This assumes separability of the trader's functionality with respect to the two types of customers, which is a realistic assumption in the context of open information markets.
2. We assumed LEN properties of the model for the simplicity of the exposition. We recognise however that the particular functional forms used will also be influenced by the queuing process at the trader. In particular, the cost function (9) will be influenced by the installed capacity of the trader. This, along with the pricing scheme, will determine the load on the system and the QoS (this in turn will influence the probability distribution function).
3. A wider range of information asymmetry situations emerge within the scope of the trading community. For example, ex-ante agency problems between an owner of a firm (who wants to purchase a trader service) and a trader service provider; different internal agency problems associated with a trader operating in an organisation (especially a large one); and last, but not least, agency problems which arise between SRs and SPs after the trading function has been carried out.
4. Finally, it is interesting to comment on the extent to which trader owners can exploit previous informational advantage. This depends on the characteristics of the owner (e.g. opportunism), but, more importantly, on environmental factors, e.g. the number of competitors. It is important to note that, if there are more competitors, agency problems are less severe.

CHAPTER 6

A Business Contract Architecture Including Quality of Service

In this chapter we identify those components of an ODS architecture (ODS-A) which are needed to support ODS enterprise requirements that have been highlighted in the previous chapters. More specifically, the chapter focuses on the development of a *business contract framework* (BCF) and the corresponding *business contract architecture* (BCA) which extend the capabilities of current ODS architectures towards support for more coherent inter-organisational electronic business transactions. This problem domain is complementary to the one in the previous chapter. Namely, once a quantitative model for designing contracts is developed, it becomes necessary to provide an appropriate architectural support for the concept of contract (or more specifically, business operations associated with contracts) in an ODS. Bearing in mind the role of contracts (as extensively discussed in the previous chapter), such a framework would alleviate information asymmetry problems and uncertainty of service provision where they may potentially arise and ultimately increase confidence in using ODSs for inter-organisational business dealings.

We first outline basic concepts of contracts from economic, legal and business perspectives

with the aim of establishing a sound base for positioning contracts in an ODS-A (section 6.2). In particular we address the economic role of contracts, the related semantic issues and a need for legal support in the form of business contract law. These concepts will then be used to develop a generic BCF, without reference to a specific business or IT scenario (section 6.3) and then to identify fundamental components of the corresponding BCA (section 6.4). In section 6.5 we will show how such an architecture can be related to a generic ODS-A, which encompasses most of the concepts of the commercially available ODS platforms. Following this, in section 6.6 we will illustrate a realisation of the concepts of our BCA based on the use of a specific ODS-A, the OMG Common Object Request Broker Architecture (CORBA). In section 6.7 we discuss the architectural notion of the QoS through which business contracts specification can be related to the underlying resource aspects. The chapter concludes with some arising comments (section 6.8).

6.1 Introduction

The combination of object oriented technologies and distributed computing represents a suitable technological base onto which a new vision for business can be built: the distributed enterprise [132]. This encompasses electronic support for intra-organisational business procedures and also inter-organisational business interactions. In section 3.1. we identified two important enterprise-related objectives, namely *i)* support for sharing and utilisation of information services and computing resources across organisational boundaries, and *ii)* the dissemination of the benefits of distributed systems to a wider user community. ODSs can play an important role in advancing the enterprise goals of their owners and users within both of these domains. Additionally, ODSs can become a new source of advantage for businesses of *all types*, ranging from small business to large multinational organisations. To this end, common business requirements need to be supported and the fundamental concepts of business practices need to be incorporated into a supporting ODS-A.

However, it should be emphasized that until recently, electronic business operations have been the privilege of large companies which had the knowledge, technology and sufficient capital to invest in an electronic infrastructure that supports business transactions

(e.g. a private network). Examples are financial institutions, airline carriers, some manufacturing companies and large retail distributors. Although Electronic Data Interchange (EDI) standards[1] have provided an additional impetus for increased electronic business transactions over the last several years, they have been mainly adopted within large organisations.

The rapid penetration of new technologies, in particular the Internet as a transport facility, and new open distributed environments[2] as a source of new services (implemented irrespective of the underlying technologies), has revealed a major concern: *the lack of a consistent architectural support for business contracts*. This is evident from the following.

1. In spite of increasing acceptance of EDI (it is estimated that between 30-40 thousand organisations have implemented some EDI features [57]), the rate of its adoption is still relatively slow. This can be attributed to the fact that in setting up EDI relationships, electronic transaction users often need to clarify the rules governing their relationship and they enter 'trading partner agreements' which are *paper* contract documents [153]. This frequently involves a lengthy process of agreement on several issues; one being the types of the EDI messages which will subsequently be used. As a result, entry barriers for use of EDI are still high [19].
2. Presently, there is the absence of a *legally valid* framework which would support all types of business dealings, both for simple electronic transactions and also for more complex inter-organisational business dealings, including international contracting [81]. This is manifested, for example, in concerns that, from a legal standpoint, new technologies such as digital signatures need to be tested in the evidentiary process as discussed in [24]. In general, it can be said that the law applicable to electronic commerce is in some ways uncertain [153]. Due to this problem (as well as a number of security issues) some commercial products still rely on a certain level of non-automated procedures. For example, an electronic commerce mechanism has been recently established for the Internet, which provides transaction, payment tracking and information delivery services for businesses and individuals [48]. However, as part of an initial procedure related to assigning personal identification numbers which

1. They cover the communication of business information in a standardised electronic form.
2. In the commercial world, these are frequently referred to as 'middleware'.

will be sent over the Internet, it is required for security reasons that credit card numbers be sent by non-electronic means to the service provider for the purpose of establishing correspondence between the two.

3. The present focus of business process modelling is predominantly on business processes within organisations (internal operations[3]). However, an important new characteristic of ODSs is that they will facilitate business interactions *across organisational boundaries* (external operations). To the best of our knowledge inter-organisational issues have not been extensively addressed in current business process modelling paradigms. Yet, today's trend toward increased global interdependencies between businesses requires a modern electronic infrastructure to support these transactions. Such an infrastructure is an important factor which will further augment interaction between organisations, e.g. those that reflect the formation of strategic alliances and the establishment of network organisations (as mentioned in subsection 3.6.1)

The technologies mentioned above (i.e. the Internet and ODSs) can be regarded a suitable initial base for the development of that part of an ODS architecture which should provide electronic support for business activities of enterprises. However, there are still a number of problems that need to be resolved to provide a coherent support for the full scope of business operations within and among enterprises. One important ingredient that is still missing is electronic support for business contract operations such as contract negotiation, fast establishment of contractual agreements and the operations associated with contract performance. In the following we will attempt to outline those technologies available today that can be used to develop a business contract architecture (BCA). Three broad categories of architectural components can be used to develop such an architecture, as depicted in Fig. 6.1. (this separation is initially proposed in [154] and further developed in [95]).

- Computing and communications *technology domain*, which can be regarded as a base for end-to-end electronic transfer of messages and electronic access to information. This includes networks such as value added networks (VANs) and the Internet, as well as protocols such as those based on X.400 and X.500 standards. Additionally,

3. These operations embrace for example, a set of business processes needed to market, develop, manufacture, distribute and sell products and to administer and manage the operation itself [110].

some fundamental mechanisms of an ODS-A, such as type management systems, can belong to this category: they can provide a basis for a common understanding of information and service types (as will be elaborated in section 6.5). A type management system for example can be used to store various types of EDI messages and documents, which are described in the following.

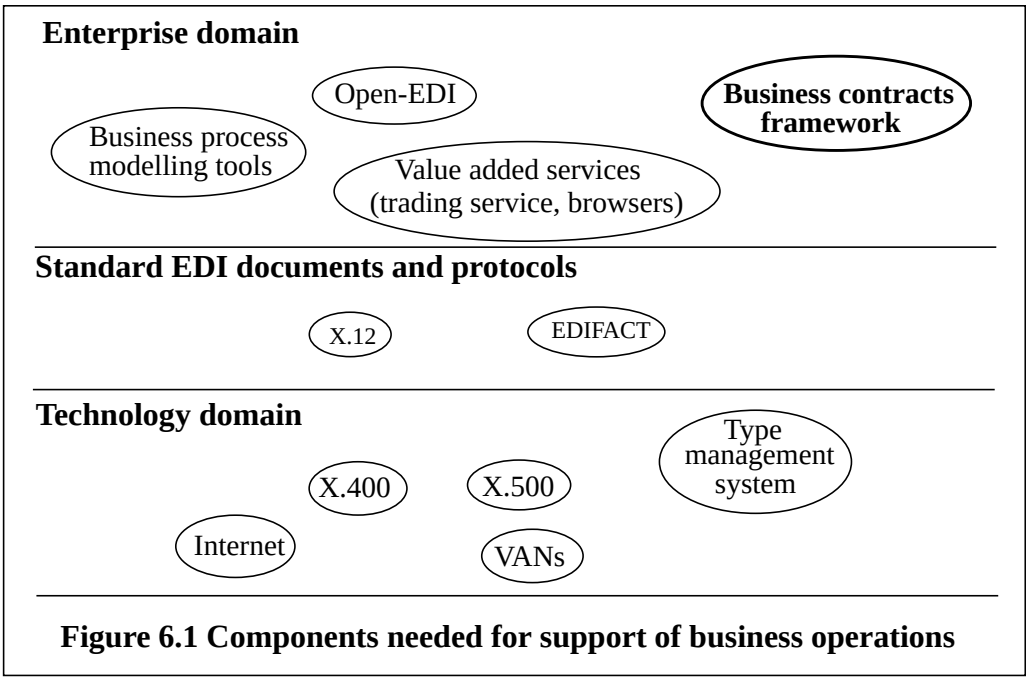

Figure 6.1 Components needed for support of business operations

- Established *EDI documents and protocols*, which provide a common understanding of elementary messages that are used to facilitate exchange of business information between enterprises. EDI standards can be regarded as an initial attempt to support some basic inter-organisational business transactions. At the international level, one such standard is set under the auspices of the United Nations, and is called EDIFACT: EDI for Administration, Commerce and Trade [146]. In the United States, the EDI standard is developed by American National Standards Institute (ANSI) within the task group X.12[4]. EDI standards facilitate the exchange of documents in standardised electronic form and are presently gaining wide acceptance in various industry sectors [148], [154]. While these standards provide a basis for common understanding of messages to be exchanged between businesses, they support only limited semantics related to business interactions and relationships. Having recognised that this is a

4. Work has recently started to align these two standards and this can be regarded as another effort to further facilitate EDI at international level.

major obstacle for the rapid introduction of EDI, ISO has initiated work towards addressing these semantic issues. For example, the initiative known as *open-edi* [69], is targeted towards the specification of standard business scenarios that can be employed without prior trader partner agreements [19]. The scope of the open-edi belongs to the enterprise domain of electronic business dealings: the subject of the next category of architectural concepts.

- *Enterprise domain,* which deals with the typical business activities and relationships within and among organisations[5]. This includes a number of procedures which can be standardised (and are thus suitable to be automated), and also business specific details. We anticipate that a number of generic value added services will be included in this domain such as services based on the RM-ODP trader and World Wide Web browsers [14]. This is the category within which various *business process modelling* efforts belong. An example is a recent initiative within the OMG consortium, the establishment of the Business Object Management Special Interest Group (BOMSIG) [110]. This group aims to identify a minimal set of business objects which are common to different business process modelling tools but which are also generic to many businesses. Other examples include Digital's Framework Based Environment and Object-Oriented Business Engineering by Open Engineering Incorporated [132]. Despite the initial success of many of these methodologies, it can be said that little work has been done to address the full scope of inter-organisational electronic business integration; this indeed being a major goal of ODSs. The existence of a comprehensive *business contract framework* (BCF) which should provide support for a wide range of contract operations can be regarded as one of the major ingredients for the acceleration of interactions among enterprises.

It is this last category of architectural components that is the scope of this chapter: we are striving to develop of a framework which will support one important business concept: the *business contract*. Such a framework should meet the following requirements.

1. It should be *generic* enough to be applied to any ODS architecture. This emanates

5. We note that the basic set of business concepts which have been identified within the RM-ODP enterprise viewpoint (as outlined in subsection 2.5.1) need to be further extended to cover specific business scenarios.

from the requirement for interworking between ODSs, as discussed in section 2.2.

2. It needs to incorporate the *semantic* of contracts. This is needed to promote common representation of business contract operations; an aspect which indeed the EDI community has recognised as important for the shortening of the negotiation process that is currently used to reach an agreement on messages and protocols to be used for subsequent EDI interactions.
3. It should have the capability to include and reflect major *legal rules* of domains within which ODS will be implemented. We argue that by including relevant legal rules, a BCF can contribute towards increased confidence in the use of ODSs for a great deal of business interactions between enterprises.

We argue that an architectural framework which meets the above requirements should be based on a sound understanding of the different aspects of contracts that are used in the real world. This includes the economic, legal and business standpoints. To this end, in the following section we will elaborate the basic concepts of contracts from these three perspectives.

6.2 Contracts: different perspectives

Contracts[6] are common phenomena in every day life. They have the purpose of facilitating some exchange or regulating one of many interactions between people in economies and societies which are based on labour specialisation. Generally, a contract can be defined as an agreement entered into voluntarily between two or more parties who promise to exchange money, goods or services according to a specified schedule. This is the promise enforcable by law [45]. In simple exchanges (e.g purchases), contracts are not necessary. However, more complicated and time consuming transactions call for an agreement consisting of reciprocal promises that form the contract and that will remain in force until all parties are satisfied. If one of the parties fails to keep the promise, the other is entitled to legal recourse against him [45]. Enforcement of good faith in matters of contract is

6. Although the scope of our study covers business contracts the text in this section can be applied to a number of general contract concepts.

considered among the most important functions of legal justice.

Contracts are of concern for economics and legal sciences, but also for everyday business dealings. Since ODSs will facilitate the execution of a wide range of electronic business transactions, it is of crucial importance that business contracts be supported within an ODS-A. Therefore, we will look at contracts from economic, legal and then business perspectives with the aim of appropriately positioning them within a supporting ODS-A. We will also consider novel contract issues that are introduced by electronic business dealings.

6.2.1 Economic perspective

The understanding of contracts from an economic standpoint can be used to:

- clarify the *purpose* of contracts
- explain different *kinds* of contracts appropriate for different types of inter-organisational transactions
- provide more insight into the contract *negotiation* process.

From an economic point of view, contracts arise as a result of efficiency-seeking behaviour in a world of limited information [114]. They are created to minimise transaction costs between specialised factors of production. They also involve issues of higher complexity than simple market transactions (e.g. buying and selling).

Contracts involve some kind of continuing relationship between two or more parties [140]. They specify payments, which determine each party's incentives, including incentives to fulfil obligations (e.g. of agents) and to exercise their rights (e.g. of principals). In the previous chapter we have seen how agency theory can be used to design optimal contracts which provide trade-offs between incentives and the sharing of exogenous risks in the presence of uncertainty.

The investigation of contracts has recently become one of the dominant subjects of re-

search in economics [150][7]. It is now recognised that different kinds of contracts can explain different types of economic governance structure within which commercial transactions take place [151]. These contract types are determined by three critical dimensions of contracts (Fig.6.2), namely uncertainty, asset specificity and the frequency with which transactions recur, as described in subsection 3.4.2. Considering different combinations of these dimensions, several examples of commercial contracts are illustrated in Table 6.1, based on [151]. The table includes the frequency and asset specificity dimensions of transactions as follows[8].

- Three frequency categories are identified: one-time, occasional and recurrent transactions. Since few transactions are of one-time type[9], it is sufficient to consider only occasional and recurrent transactions.
- With regard to the asset specificity, non-specific investments refer to standardised products. These involve few risks since buyers can easily turn to alternative sources and the supplier can sell output intended for one order to other buyers. On the contrary, specific investments refer to non-deployable assets specific to the relationship between parties, e.g. cost of capital and human specific investments related to particular activities.

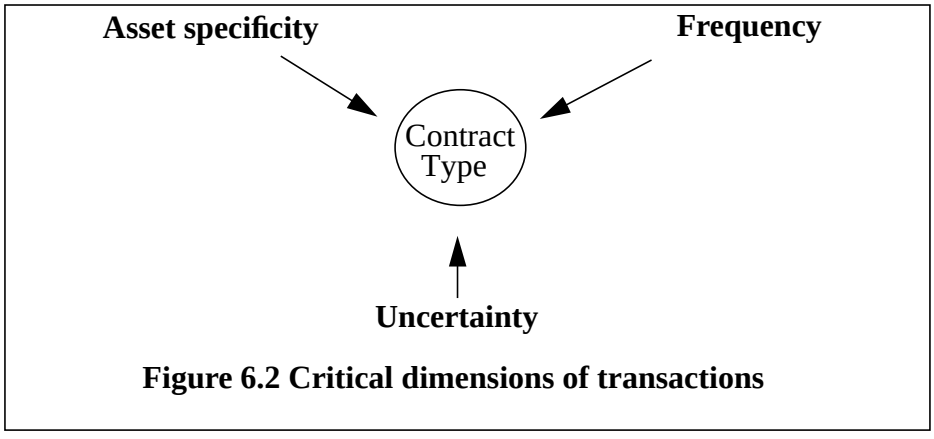

Figure 6.2 Critical dimensions of transactions

Our aim is to apply this analysis to services in the context of ODSs. Based on the charac-

7. Contracts have for long time been a concern of business people and lawyers. It is only in last 10-15 years that they have attracted the attention of economists. As a result a special branch of economics called *contract economics* has emerged.
8. The third factor (uncertainty) is taken as the parameter, and its influence will be separately discussed, as in [151]
9. e.g. purchasing local spirits from a shopkeeper in a remote area of a foreign country.

teristics of electronic business transactions, it would be beneficial to identify the most appropriate kind of contract which would allow firms to exploit the benefits of an ODS for more efficient structuring of their business interactions. Such a contract type will further influence the appropriate *governance structures* within which business transactions should take place. It will also influence the corresponding *legal framework:* the related categories of contract law[10].

Table 6.1: Sample commercial transactions

		Investment Characteristics		
		Non-specific	Mixed	Specific
Frequency	Occasional	purchasing standard equipment	purchasing customised equipment	constructing a plant
	Recurrent	purchasing standard material	purchasing customised material	site specific transfer of a product

Various governance structures impose different architectural requirements, while the related categories of contract laws can elucidate the legal requirements needed to support different commercial contracting interactions which will take place within an ODS.

The governance structures and the corresponding categories of contract law are discussed as follows (and summarised in Table 6.2 [151]).

- *Market governance* is the main governance structure for non-specific transactions of both recurrent and occasional natures. In the former, both parties to the contract can select an alternative purchase since a product is standardised and easily obtainable from others. In the latter, they can rely on rating services or the experience of other buyers of the same product (as they are less able to count on direct experience to safeguard against opportunism). Non-specific transactions take place within a legal framework of *classical contract law,* which lays the emphasis on legal rules, formal documents and self-liquidating transactions.

10. Three broad categories of contract law can be distinguished, based on different approaches used to achieve the purpose of facilitating exchange, as developed in [151].

- *Trilateral governance* can be applied to occasional transactions of mixed or highly specific nature. Once parties to such transactions have entered into a contract, there are strong incentives to complete the contract, because specific investments have been made. Since market governance is inadequate and since setting up a transaction specific governance cannot be justified for occasional transactions, the assistance of a trusted third party, an *arbitrator,* can be used in resolving disputes and evaluating performance. This corresponds to *neoclassical contracting*, in which the assistance of an arbitrator can provide more flexibility than litigation[11].
- *Transaction-specific governance* structures are suited to recurring transactions of a mixed and highly specific kind. Their recurrent nature allows the cost of establishing specialised governance structures to be recovered. Two types of these structures can be identified: *i)* a unified governance in which the transaction is removed from the market and organised within a firm, subject to some authority relation (vertical integration) and *ii)* a bilateral governance in which the parties can maintain their autonomy. The latter structures have recently received attention as they occur in the form of network organisations (as described in subsection 3.6.1), joint ventures and strategic alliances. Bilateral governance is characterised by so called *relational contracting* in which the basis for agreement is not the individual transaction but the entire relationship between the parties as it has developed through time[12].

These contract categories can be used as a starting point to identify fundamental economic contract categories which should be appropriately represented and stored within an ODS (as will be elaborated in subsection 6.3.1). In relation to the influence of the third critical dimension of transactions, *uncertainty*, the following can be said [151].

- An increased level of uncertainty does not influence non-specific transactions as their continuity has little value since new trading relations can be easily arranged. Competitive forces provide self-regulation which alleviates possible opportunism. Thus classic contract law applies no matter how uncertain the purchase is.

11. Since it is recognised that the world is complex, that agreements are incomplete and that some contracts will never be finished unless parties have confidence in the settlement machinery.
12. These contracting issues are for example the subject of a great deal of corporate law.

- An increased level of uncertainty has an impact on transaction-specific investments, since contractual gaps will be larger and the occasions for sequential adaptations will increase in number and importance. One solution is to sacrifice a valued design feature and adopt a more standardised product. Other alternatives are to devise an elaborated governance structure for occasional, nonstandard transactions, or to place them under the unified governance for recurrent transactions.

Table 6.2: Governance structures and commercial transactions relationship

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">Investment Characteristics</td></tr>
<tr><td>Nonspecific</td><td>Mixed</td><td>Specific</td></tr>
<tr><td rowspan="3">Frequency</td><td>Occasional</td><td rowspan="3">Market Governance
(Classical Contracting)</td><td colspan="2">Trilateral Governance
(Neoclassical Contracting)</td></tr>
<tr><td rowspan="2">Recurrent</td><td>Bilateral
Governance</td><td>Unified
Governance</td></tr>
<tr><td colspan="2">(Relational Contracts)</td></tr>
</table>

In addition to identifying different kinds of contracts, economics can provide more insight into the *contract negotiation* process [125]. Negotiation is a ‘dynamic process of adjustment by which two (or more) parties, each with their own objectives, confer together to reach a mutually satisfying agreement on a matter of common interest’ [90]. Contract negotiation is an area where contract design topics are important and where one considers the influence of uncertainty, asymmetric information and opportunism on designing optimal contracts, as extensively studied in chapter 5. Once a contract is negotiated, it can be submitted by any party to a legal expert or authority for validation.

A ‘real-world’ picture of contracts can be completed by incorporating the legal and business perspectives, as will be discussed in the following two subsections.

6.2.2 Legal perspective: general theoretical considerations

From a legal standpoint, a contract alleviates mistrust in a world of uncertainty, by con-

straining the unpredictable activities of the other autonomous parties [101].

Behind any system of contract law is a corresponding *legal contract theory*. Legal contract theories tell authoritative decision makers how to regulate contracting behaviour. Legal theories have a process aspect and a substantive aspect [131].

The *process* aspect of a legal theory supplies criteria for dividing regulatory authority among legal institutions. For example, a legal theory could hold that courts should enforce contracts but that the legislature should develop a disclosure scheme such that people do not contract on misleading information [131].

The *substantive* aspect tells decision makers how to regulate. There are two important legal contract theories:

- the 'traditional theory' (more popular with lawyers and courts), which tells decision makers to supply *fair* terms
- 'law and economics theory' (more popular with academics) [131], which holds that decision makers should not be guided directly by fairness considerations; rather they should be guided by providing *efficient* terms.

It is recognised that there is still much to be discovered in the field of substantive aspects of legal contract theories in order to explain and predict different contractual relationships which occur in real life, for example, how to regulate contracting behaviour in cases of incomplete contracts. This probably explains why courts rarely engage in completing contracts [131]. These issues are however beyond the scope of this thesis.

In subsection 6.3.2 we will outline how some elements of legal contract theories can be used in establishing a BCF.

The economic and legal aspects of contracts are institutionalised in business contract law. Since business contract law relates closely to business practices, we discuss related legal issues under the following heading of the business contract perspective.

6.2.3 Business perspective: business contract law

Business contracts are an essential part of business dealings. A business contract is an agreement between two or more parties to undertake or not to undertake particular business activities.

To understand business contracts, with the aim of providing electronic support for these in an ODS, the following concepts are important to consider [101].

1. Contract *domain*. To prevent any ambiguity, contracts are bound to a particular contract domain, such as a state or a country. The rules and policies with which members of a particular domain comply normally emanate from statutory or administrative law.

   In the real world, there are many autonomous countries, sometimes composed of autonomous states or provinces. Each of these has its own legislation which might place restrictions on the contracts that can be made within their boundaries, which define the contract domain. Typically, the legal system of these countries will enforce only the contracts deemed valid by their legislation (through courts).

   A contract domain is not necessarily restricted to country or other regional authorities. For example, many professional bodies are responsible for maintaining a register of licensed practitioners in their profession, and these professional bodies can regulate and enforce appropriate behaviour of those licensed practitioners [101]. Similarly, in the world of commerce, certain rules and policies are applied to govern specific fields of commerce, such as stock market trading rules, as will be shown in subsection 6.4.2.

   Interactions between parties belonging to different contract domains are normally governed by a set of policies which apply across domains, and thus define a higher-level domain. An example of a set of rules and policies which define such a domain is the United States Uniform Commercial Code, which is a uniform statute adopted in whole or in part by each state legislature in the US to govern specified fields of commerce [120]. Therefore, a *contract domain* defines the scope or boundary in which a contract is deemed to be valid, a contract dispute can be arbitrated and correct contract behaviour can be enforced [101].

2. Contract *validity.* To be valid, a business contract must be aligned with the rules incorporated in the business law of the specific contract domain. In general, a contract

must include the following elements [120], [141]:

a) An *agreement*: an offer seriously and clearly made by one party to another party, who must accept it seriously and clearly and without reservation. In addition, both parties make the agreement voluntarily, without restraint or influence, acting of their own free will.

b) *Consideration*: This is something of value that each party gets or gives. Each party thus establishes an obligation to each other. With few exceptions it must be shown that both parties intended to bargain and have actually exchanged something for a contract to be enforceable by a court. Consideration can take the form of money, or an act performed or withheld, services rendered, other property or individual rights. In general it need not be tangible or posses an economic value.

c) *Competence* (or capacity): The ability to incur liability (debt) or to gain legal rights (e.g. a person who is insane or below a certain age might not be really competent to make a contract).

d) *A legal purpose*: A contract cannot be enforced unless the actions agreed upon are legal in the jurisdiction where the contract is made (in other words a contract's purpose or object must comply with law). One party cannot bring suit against the other for breach of contract if the act required by the agreement is illegal.

A court will enforce a contract if it meets the four requirements. In general, enforcement of contracts will occur only if the contract is breached by a party to that contract. These four elements provide (legal) binding on both parties.

3. Contract *monitoring*. Although a contract is intended to regulate the activities of the parties involved, reality can often fall short of expectations. Having committed to a contract, a business wishes to ensure that it receives the value it expected from the contract. For this reason, the business might monitor the activities that are governed by the contract to ensure that the other parties comply with the agreed terms. Equally, the business might monitor its own activities to ensure that other parties have no cause for complaint. For example, a business might establish procedures to scrutinise all invoices to check that they receive the discounts agreed in their contract. Some methods of contract monitoring might be carried out irregularly: an example is customer satisfaction surveys [101].

4. Contract *enforcement*. As a consequence of contract monitoring, an organisation might decide that the contract has not been honoured by another party and might wish to take corrective action. This process of contract enforcement can take many forms (possibly more than one):

   a) to require the other party to conform in the future to the contract (e.g. to ensure that goods are delivered on-time)

   b) to require the other party to correct the previous problems (e.g. to replace damaged merchandise)

   c) to demand compensation for past problems (e.g. to pay interest on outstanding balances)

   d) to terminate the contract.

   Contract enforcement might occur through direct discussion between the contract participants. If this does not produce a satisfactory resolution, the dispute can escalate through various levels of mediation or arbitration, ultimately leading to a court case.

6.2.4 Technology perspective

New services which are made possible through high speed networks and distributed computing bring novel aspects to business contracts that have not been known to the business community of the past. Examples of such services are electronic banking, ordering and trading. These bring new advantages, but also new concerns for business transactions, such as an increased possibility of fraud (so that complex encryption methods are needed). In addition, increased electronic integration between enterprises requires an extension of current business law in order to ensure the legality of such transactions (e.g. assuring legality of electronic signatures). These new features need to be considered when developing a BCF.

Therefore, in establishing a BCF, one should take into account economic, legal and business aspects of contracts. Additionally, the IT itself contributes to new characteristics of business contracts, and this perspective should also be taken into account. Fig. 6.3 depicts different contract perspectives which need to be accounted for when developing a BCF.

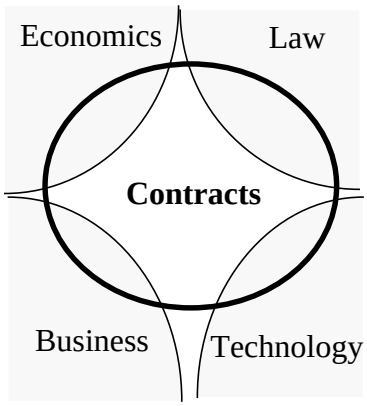

Figure 6.3 Various perspectives of contracts

It will be now illustrated how we have used these four perspectives of contracts to develop our BCF model.

6.3 Development of a Business Contract Framework

Observing how contracts operate in the real world establishes the basis for identifying the relevant concepts of a BCF. We note that the BCF developed in this section can be related to many business or IT scenarios. Such a framework can then be used as an extension of a core architecture of an ODS, but also in other IT domains, such as the Internet [95].

We now start with the identification of fundamental concepts of our BCF. In the course of the development of this BCF, we will follow relevant concepts from the RM-ODP.

In the previous section we have introduced a definition of a contract, which includes economic and legal notions. In the RM-ODP the contract is defined as ‘an agreement governing part of the collective behaviour of a set of objects’ [13] [65]. Further, ‘A contract places obligations on the objects involved. An obligation is a prescription that a particular behaviour is required. An obligation is fulfilled by the occurrence of the prescribed behaviour’.

13. Object is defined as a model of entity. An object is characterised by its behaviour and state.

This definition is general enough to cover any kind of contract. Our analysis in the previous section provides a more complete picture, with the aim of arriving at a generic BCF which can encompass many concepts typically used in the operations associated with business contracts.

The typical ‘life-time’ of a contract in our BCF includes:

- contract establishment, which includes contract negotiation and validation procedures
- contract performance, which is related to the performance (behaviour) of parties to the contract during the period of contract validity
- post-contract stage.

This separation is similar to the concepts defined in the RM-ODP [65], associated with contractual behaviour, which consists of the following:

- establishing behaviour
- enabled behaviour
- terminating behaviour.

While there is a variety of different kinds of contracts that govern specific relationships between trading partners, a large class of contracts are also standardised to reduce the cost of setting up the contract agreement. This, along with the fact that there are a number of elements which are common to many contracts, served as our motivation to introduce the concept of *contract template* in our BCF. According to [65], <X> Template is ‘the specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it. <X> can be anything that has a type’.

A typical contract template can include *contract elements* such as:

- the roles of the parties
- the period of the contract (the times at which the contract is in force)
- the nature of consideration (what is given or received), e.g. actions or items

- the obligations associated with each role, expressed in terms of criteria over the considerations (how much, how many, how often), e.g. quality, quantity, cost and time
- the domain of the contract (which determines the rules under which the validity, correctness, and enforcement of the contract will operate).

A contract template thus contains certain *semantics* of a business contract. These semantics can represent different scenarios associated with business contracting. Additionally, there can be certain *relationships* between different kinds of contracts. For example, a general contract template may require further refinement to specialise the template into a specific business contract.

This description of a contract template is similar to the corresponding RM-ODP definition, which states that the specification of contract may include [65]:

- 'a specification of the different roles that objects involved in the contract may assume, and the interfaces associated with the roles
- QoS attributes, where QoS is defined as 'a set of quality requirements on the collective behaviour of one or more objects' (QoS may be specified in a contract or measured and reported after the event)
- indications of duration of periods of validity
- indications of behaviour which invalidates the contract
- liveness and safety conditions.'

We now turn to the identification of other concepts drawn from the economic, legal and business perspectives, that are needed to build our BCF.

6.3.1 Economic perspective

The findings from economics can be used to identify different *kinds* (or *types)* of contracts which reflect specific economic or business circumstances, as follows.

- *Standard* contracts, which cover non-specific (i.e. standardised) transactions with a low level of uncertainty (irrespective whether they are of recurrent or occasional type, as explained in subsection 6.2.1). Such contracts typically include a small number of well defined contract elements. An example can be a house purchase contract, which in its basic form may be regarded as standard in many jurisdictions. In this case, the structure of the corresponding contract templates is fixed and contract instances will represent actualisation of the contract elements. Additionally, certain simple contracts can be customised, in which case subspecialisation of contract templates is required.

  Contracts in this category are normally executed within market governance and thus classical contract law can be applied. The main concern here is checking the validity of contracts and contract enforcement (more legally oriented operations). For example, an architecture which is based on this BCF may include several components to support this functionality (as will be elaborated in section 6.4). These include a repository which contains the rules and polices of the corresponding legal system, and those components which ensure the legal validity of a contract and correct behaviour of the parties included in the contract (we will refer in section 6.4 to these components as legal rules repository, contract validator, contract monitor and contract enforcer). It is worth noting that many validation aspects can be implicitly incorporated within the underlying contract templates.

- *Non-standard* contracts which govern interactions that involve *occasional* transactions. These kinds of contracts should be guided by neo-classical contract law and thus a third-party arbitrator can be used. Its functionality can be implemented by an appropriate architectural component (i.e *contract arbitrator)* which will be introduced in section 6.4.

- *Non-standard* contracts which govern interactions that involve *recurrent* transactions: they specify long-term relationships between parties to the contract. Relational contracting is applicable to these contract types.

  We envisage that for this kind of contracts the use of a BCF will be reduced to the necessary checking of contracts during the negotiation phase and to certain contract monitoring procedures, but the use of enforcing mechanisms would be minimised. Since the parties' relationship is based on trust and since the cost of settling claims

can be unjustifiable, the parties to the contract have incentives to maintain their own relationship and, if required, adjust contractual terms without involving an enforcing mechanism. An example of such a relationship is an electronic integration between an insurance carrier and insurance agent, as presented in subsection 3.4.3.

To store different contract templates within a BCF, and represent different relationships between them (e.g. subtype[14] or substitutability), an appropriate repository, the *contract repository*, is needed. This can be publicly accessible, private, or third-party owned. For example, standard contract types can be stored in a repository which can be accessible publicly, or according to certain permission rules. We anticipate that the availability of standard contract types would shorten the negotiation time needed for an agreement between companies and would thus promote electronic business interactions between enterprises. However, other contract types, which are more suitable to cover transactions of a specific nature can be stored within a specific community, perhaps private (or a third party) repository.

In addition to being a storing facility, a contract repository should facilitate the manipulations of the contract templates. This can be particularly useful in selecting contract templates and supporting contract negotiation, as will be shown in the following subsections.

The economic aspects of contracts are also relevant when considering the *contract negotiation* process and designing efficient contracts. Contract negotiation is a multistep process in which parties with conflicting interests come to a mutual assent through direct interaction. Obviously, this process is dependent on the environment and the parties involved and belongs to the application domain. However, in the case of limited knowledge or information available about the environment and/or the other parties a third-party negotiator can be introduced. This party has the mediation role since it serves as a trusted party which, because it possesses information about the negotiating environment and knowledge about the negotiating parties, can reduce uncertainty about the outcome of the negotiation and thus provide a more efficient negotiation process. We note that QoS negotiations represent a subset of contract negotiation procedures.

14. A type A is a subtype of a type B, if every <X> which satisfies A also satisfies B [65].

Once the contract instances are agreed and instantiated, they may need to be kept for future auditing or monitoring purposes.

Contract negotiation is the first phase of a ‘contract life-time’. It is followed by other stages, such as contract validation, monitoring, and enforcement, which are discussed in the following subsections.

6.3.2 Legal perspective

We have also used findings from legal contract theories in the development of a part of our BCF.

The substantive aspect is reflected by a set of prescribed *rules* which reflect a legal system and which need to be accessible to a BCF (e.g. rules associated with contract validity and contract enforcement)[15]. In developing the BCF we will assume that the terms which regulate contracting behaviour are given *a priori* (e.g. enacted by regulatory bodies such as state legislatures).

The process aspect of a legal contract theory can be used for *structuring* purposes, to define legally related roles in a BCF and the relationships among them. In relation to this, our BCF makes provision for *i)* an access to the legal rules repository of the corresponding jurisdiction domain, so that a legally allowed and valid behaviour of the parties to the contract can be ensured and *ii)* a contract enforcing role (e.g. a role of a court).

We note that there may be a number of additional supportive roles and these will be described in more detail in section 6.4.

15. These rules can be stored in the legal rules repository, mentioned in the previous subsection.

6.3.3 Business perspective

From a business perspective, a BCF should be flexible enough to support a wide range of business dealings of enterprises that are governed by rules and policies of a particular jurisdiction. In legal terms, a jurisdiction determines a contract domain (recall subsection 6.2.3).

A *contract domain* must define:

- a set of contract validity rules; as discussed previously, these rules can be stored in a legal rules repository
- a procedure to arbitrate contract disputes
- a procedure to enforce contract behaviour.

The notion of a contract domain can be mapped onto the wider concept of domain in RM-ODP. First, we note that in RM-ODP an <X> domain is defined as a set of objects each of which is related by a characterising relationship <X> to a controlling object[16]. Generally, the controlling object is not a member of the associated domain. In the case of a contract domain, a characterising relationship can be 'legally validated contract by' a contract legality object.

Contract validation is the process of ensuring that a contract satisfies the contract validity rules of the nominated contract domain. In some contract domains, the contract validity rules require that the parties must be members of the domain. Contract validation involves multiple activities:

- checking the *capacity* of parties, including verification of a party's own capacity and checking capacity of other parties
- checking the *legal purpose* of the contract (whether it is legal in the corresponding jurisdiction)

16. Examples of domains mentioned in the RM-ODP are security domain, authorisation domain, management domain, addressing domain and naming domain [65].

- checking the *clarity* of contracts
- checking whether a *consideration* element of a contract is specified.

Contract monitoring is the process of observing the activities of the parties for the purpose of ensuring that those activities correspond to the contract. Contract monitoring can be performed by:

- the parties themselves
- third-party agents acting on behalf of individual parties
- trusted third-party agents acting on behalf of all participants in the contract.

Contract monitoring can be a continual process or it might occur only from time to time.

Contract enforcement is the process of ensuring actual behaviour conforms to the contract (pro-active enforcement), or by ensuring corrective actions to minimise the deviation from the contract (reactive enforcement). Corrective actions might be performed by the parties to the contract, or by some external object, a contract enforcer, at the direction of the domain. The effectiveness of contract enforcement might be limited if some of the parties are outside the contract domain (which is why domain membership might be required for a valid contract). If a party outside the domain does not comply with the enforcement, then the ultimate sanction of the domain is to exclude that party from contracts within the domain in future.

Concepts of a business contract validation, monitoring and enforcement can be regarded as a part of application domain or an extension of an ODS architecture (such as our BCF). These do not appear in the RM-ODP.

We will now turn to illustrate the concepts of the BCF identified so far with an example which closely resembles the typical contract operations within a legally valid contract framework. The aim of this example is to illustrate a temporal order of these operations and those parties which perform these activities. Note that the order of steps in this example reflects one of many possible scenarios.

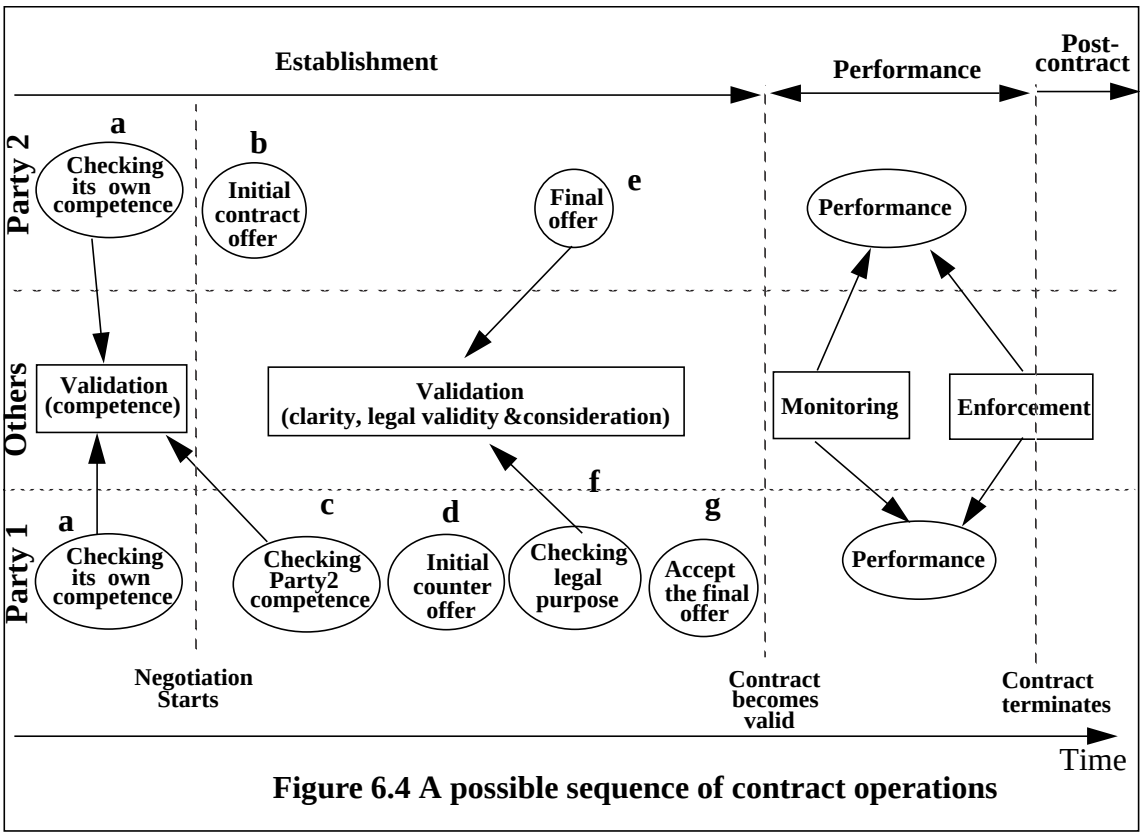

Figure 6.4 A possible sequence of contract operations

Typical contract operations: an example

We first outline the process of contract establishment (Fig. 6.4). In some cases each party may be required to check its own competence before contract negotiation commences (step a). This is similar to real world scenarios in which certain assurance credentials (e.g. licensing and endorsement) are required as the first element in establishing a contract, as also suggested in [77]. This normally means that the party has entered the contract domain.

The negotiation process starts with say party2 making an initial contract offer (step b). Before party1 seriously considers the offer, it may want to check the competence of party2, based on information about party2's identity (step c). If successful, party1 can either accept the offer or submit a counter-offer (step d). The negotiation process can proceed for example with party2 checking the validity of the counter-offer (e.g. legal validity and consideration elements, as depicted in the figure), step e. If successful, party2 can then submit a final offer. If checked for legality (step f) and accepted by party1 (step g), this contract will have a legally valid status.

The performance of the parties to the contract involves monitoring of their activities and enforcing decisions if there is non-performance to the contract. After the expiry of the contract some enforcing decisions still can be done, as also depicted in the figure (post-contract period).

6.4 A Business Contract Architecture (BCA)

Having explained the basic concepts and typical operations associated with contracts in our BCF, we now turn to the description of how these can be used to develop the core of the corresponding *business contract architecture* (BCA). Our aim is to derive such a BCA which will include a set of architectural components that can be used in many contractual arrangements[17]. We further discuss how such a BCA can be used to extend an ODS-A, so that business contract operations will be supported.

6.4.1 Main architectural components and their relationships

We will map the BCF concepts onto the roles of the corresponding BCA *components* and their relationships. In developing this BCA we will assume that an ODS-A includes a set of security components which provide support for procedures such as authentication, authorisation and encryption [77].

The architectural components identified (with their roles) are as follows (Fig. 6.5).

- *Contract repository* has the role of providing a common understanding of contractual types (e.g. checking whether a contract type and contract elements which comprise them are known to the system). It stores the following information.
  a) Some general *contract element types* which are common to many contracts (e.g. the date of the contract agreement, duration of the contract, parties to the contracts, etc.). This can, for example, include established EDI messages and documents.

17. The specifics of the applications which use such an architecture may require a number of additional components to support the specific application requirements.

b) Different *contract types*[18] as identified in the previous subsection. These can range from very simple contracts in which there is no ongoing relationship between parties (i.e. a very short contract performance period), the items being traded are well defined and have minimal complexity; to very complex contracts such as those governing many inter-organisational interactions and the most complex being international contracts.

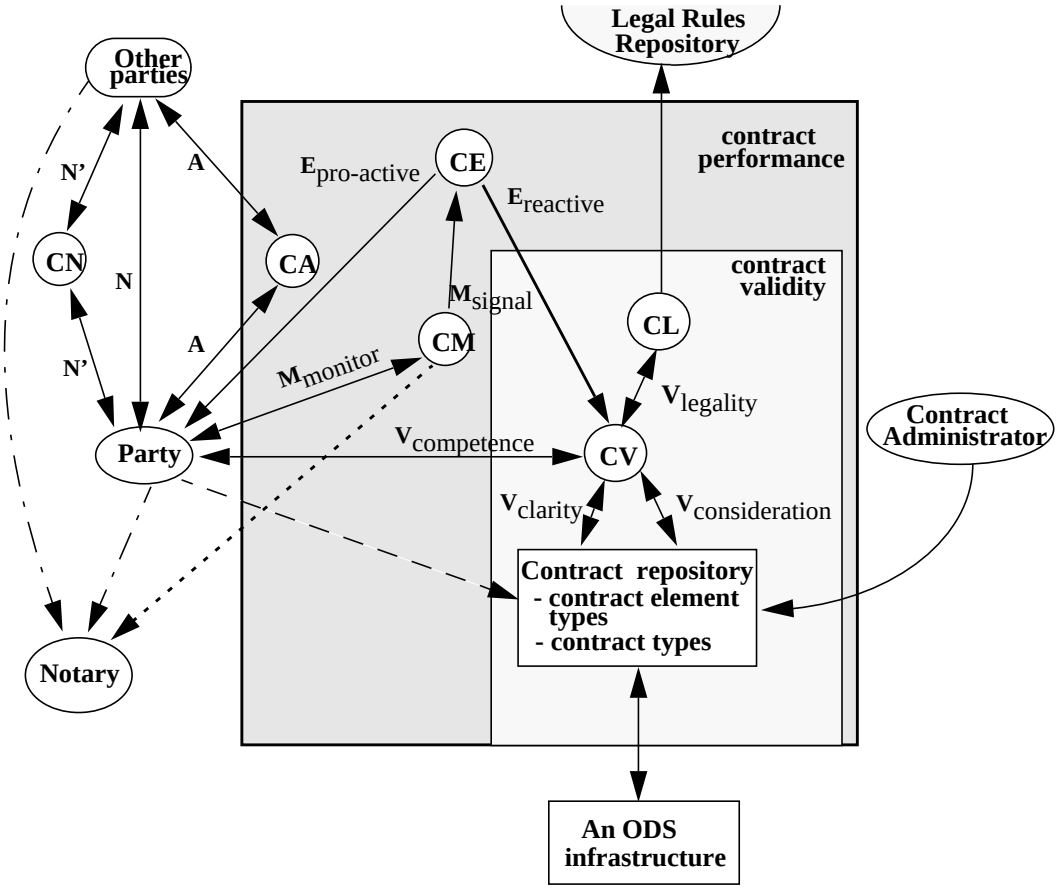

Figure 6.5 A business contract architecture (for notation see text)

The dashed line in Fig. 6.5 (between a party to the contract and the contract repository) reflects the fact that trading partners will typically access the contract repository before being involved in contractual operations. This will be normally done when searching for a desired contract type.

18. These can be represented by using the corresponding contract templates.

Descriptions of these types are introduced to the contract repository through a special trusted authority called a *contract administrator* (presumably a human).

- *Notary* has the role of storing contract instances after the contract has been agreed upon and checked for validity (dashed-and-dotted lines). This can be later used as evidence of agreement in the contract monitoring and enforcement activities (dotted lines).
- *Legal rules repository* (LRR) stores the rules and policies of a particular legislative domain.
- *Contract validator* (CV) has the role of ensuring the creation of legally valid contract instances. This includes the checking of the following aspects of contract validity.
  a) The competence aspect of contract validity ($V_{competence}$). To accomplish this, the CV should verify the capacity of parties willing to enter a contractual relationship.
  b) The clarity aspect of a contract template. To this end the CV can contact the contract repository of the BCA ($V_{clarity}$). We anticipate that in most cases the contract will be clear if it is derived from a template in the CR. The CV can be used however, to provide additional checking should a need for this arise.
  c) The legal purpose element of a contract ($V_{legality}$), based on the information in the legal rules repository. This is done through the *contract legality* (CL) object.
  d) The *consideration* element of contract ($V_{consideration}$). This can be done by checking whether the contract template contains those contract elements which describe what is exchanged between the parties.

  It is worth noting that the CV does not need to be used at each contract establishment procedure. For example, once the parties to the contract have established a contractual relationship, they both deal with legally valid contract types which may be exploited in subsequent contract negotiations or renegotiations.
- *Contract negotiator* (CN) has the role of mediating the negotiation process (relationship N'). An electronic support for negotiation brings a number of new roles for such parties, referred to as electronic brokers, as extensively discussed in [122]. Alternatively, the negotiation can be carried out by the parties themselves

(relationship N). During this stage the parties can exchange several contract templates, which represent a set of offers and counter-offers, as depicted in Fig. 6.4 and Fig.6.5. During the negotiation phase, the negotiated contract template can be submitted for validity checking, as discussed earlier.

- *Contract arbitrator* (CA) has the following roles:

  a) evaluation of the parties' performance to the contract, via monitoring and recording their behaviour (A)

  b) resolving disputes - making decisions which are beneficial for both parties and providing feedback.

  Recall that this component is needed only for non-standard contracts (otherwise the Contract Monitor component is sufficient).

- *Contract monitor* (CM) has the following roles.

  a) Monitoring activities of parties, measuring performance and recording the relevant events ($M_{monitor}$). After the contract elements are checked for validity, either party may request to check whether the contractual obligations have been met during the contract realisation.

  b) Dealing with non-performance of parties: if the CM detects a contract non-performance it should signal non-performance to the contract enforcer (M_{signal}).

  The CM may not be needed since some monitoring mechanisms can be done by the parties themselves (and possibly verified later by courts if needed; similar to maintaining diaries or other records which can be regarded as some sort of legal documents).

- *Contract enforcer* has the role of making an enforcing decision, upon being signalled by the CM. This can be done:

  a) directly on the parties to ensure that the actual behaviour conforms to the contract ($E_{pro\text{-}active}$),

  b) by informing the CV which may prevent further access to the system by non-conforming parties ($E_{reactive}$).

  It is important to note that the pro-active approach to contract enforcement is not widely used in business contracts. We envisage, however that a system of electronic

commerce can effectively support pro-active enforcement, since a business law is normally less ambiguous than common law and direct interpretation is possible. Additionally, reactive and post-contract enforcement will usually require arbitration and possibly human intervention in determining the appropriate (corrective or punitive) actions [101].

We note that this analysis deals with *explicit* contracts: those that are applicable to situations in which there is some crossing of administrative boundaries and in which the trust among parties involved is such that contracts need to be introduced and governed by a set of rules emanating from a particular business law. On the other hand, *implicit* contracts are the matter of parties to the contract themselves and are within the scope of the application domain. In this case only those contractual components of the BCA which provide a common understanding of contractual terms may need to be involved. This for example may apply to situation in which there is full trust between parties (as can be the case in network organisations) or when contracts are enforced by some indirect mechanism (e.g an authority mechanisms within a company's hierarchy).

6.4.2 Examples of the use of the BCA

We shall demonstrate the applicability of the concepts of our BCA within the application domain by using the most common contractual arrangements within an electronic stock exchange. We have chosen this example since the semantics of contracts which cover various stock trading operations are relatively simple and the legal rules which govern the behaviour of the participants are well developed and straightforward. This will be followed by an example of the RM-ODP trader component. These two examples will be further refined in subsection 6.6.3 to illustrate the specifics of the use of the CORBA conformant platform in this context.

Application domain: an electronic stock exchange

This stock exchange example is loosely based on the Australian Stock Exchange (ASX)

[2]. We note that the ASX automated trading system [5] has completely replaced trading floors[19]. It has also introduced a new trading mechanism, whereby the traditional share certificates papers are increasingly being replaced with new types of electronic documents, called uncertificated shareholdings[20].

Trading on the stock exchange includes procedures for stock offer (to buy or sell shares), offer acceptance, payment for shares and transfer of the title. These operations are handled by accredited stockbrokers who are members of an exchange. When a company issues shares it normally does this through a stockbroker; similarly when an investor intends to buy (or sell) shares he/she must operate through a stockbroker. Acting on behalf of buyers and sellers, stockbrokers typically interact to settle the transactions. This is achieved through competitive bidding: most stock exchanges are auction markets in which stocks are sold to the stockbroker bidding the highest price and bought from the stockbroker offering the lowest price.

While there is a multitude of operations performed by the participants in a stock exchange we will focus on the most common procedures and the roles of the parties involved. These are outlined in the following (this description is based on [38], [121], [144]).

A company intending to float the shares must issue a special document, *prospectus* before approaching the public. Its purpose is to disclose all the relevant information about the company so that the quality of stocks offered can be appraised. Once the prospectus is accepted by an investor this will effectively represent terms of contract between an investor and the company. Typically, a company issuing shares will contact an investment banker (a stockbroker in its own rights, whose role is buying new shares and reselling them). The agreement between the two is called an *underwriting agreement*; this is the contract containing the final terms and prices of the issue.

A client (buyer or seller of the shares) who wants to trade on the exchange can do this only

19. The electronic trading of shares, carried out across all state capitals, started in 1990, making the ASX, according to some accounts the most automated stock exchange in the world [138].
20. This is also a good example to illustrate another point: how the capabilities of new technologies influence new ways of doing business, as discussed at length in section 3.10.

through a registered stockbroker. The client and the selected stockbroker will draw an *agreement* which covers the specifics of their interactions including the commission rate that the stockbroker will charge (brokerage)[21]. The client's stock *order* is placed by using a standard document which includes the following contract elements: the name of customer, the type of account, whether it is a purchase or sale, the number of shares, price indication and the name of the registered stockbroker.

The obligations and liabilities of the stockbrokers and other participants are governed by the standard contract whose terms are regulated by the exchange. Stockbrokers have the following roles:

- Competitive bidding with other stockbrokers on behalf of their clients.
- Transfer of title once a sale has been made: the seller's stockbroker should deliver stocks to the buyer's stockbroker. The transfer of title involves next steps.
    a) The stockbroker places an offer on the exchange (after receiving an order from its client) and when a sale has been made, the stockbroker will send a *confirmation* (a contract note [2]) to the client. This document shows the detail of the transaction, including the number of shares bought, the brokerage and usually the date on which the settlement is required together with the amount with which the investor's account would be debited.
    b) The seller's stockbroker will then draw up a *share transfer* (which must be on the prescribed form) which should be submitted to the seller to be signed.
    c) Finally, this form together with the *share certificate* of the seller (which is an official record of the equity of a particular shareholder in a company), must be submitted to the company, which will replace the name of the previous shares owner with the name of the buyer. This name is kept within the company's register of members.

The relationships between a client, a public company and stockbrokers in a stock exchange (e.g. the ASX) are depicted in Fig. 6.6.

21. Once accepted by both parties, this relationship is subject to the *law of agency.* In the typical exchange transaction the client is the principal and the stockbroker is the agent.

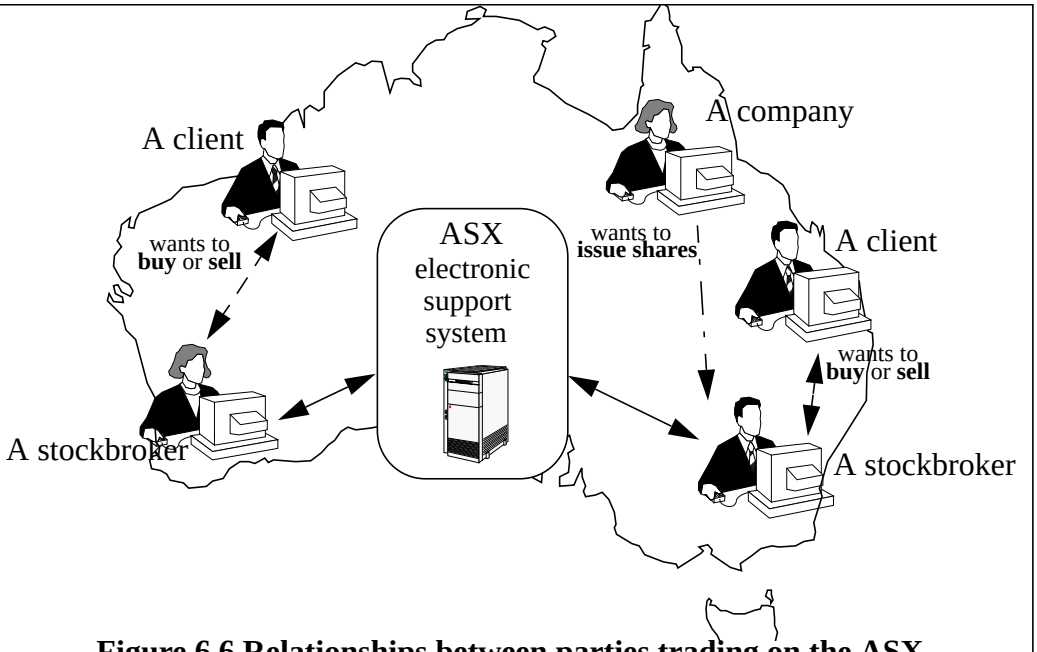

Figure 6.6 Relationships between parties trading on the ASX

We will now illustrate how these activities and roles can be described by using the concepts of our BCA.

There are different legal authorities which set rules to which all the participants to be involved in the stock trade (i.e. investors, companies, and stockbrokers) should comply: they define the *contract domain* of the exchange. For example, in the ASX there are two main bodies: *i)* the Australian Securities Commission which provides licences for accredited stockbrokers to do transactions within the ASX and *ii)* the Australian Associated Stock Exchange (AASE), which provides rules and regulations for the ASX. The latter are known as 'stock exchange listing requirements' and they specify circumstances under which a company may have its share traded on the Australian exchanges [2]. These two sets of rules and regulations constrain the behaviour of the parties who interact within the electronic exchange and can be stored either in a common exchange LRR or in separate repositories.

Some of the most commonly used stock *contract types,* which can be represented by using pre-defined contract templates, are identified below.

- *Prospectus* - a pre-defined contract template[22] which a company that issues shares will use in order to set out the terms and conditions associated with a new issue of shares. Once the prospectus is validated by the ASX, it may be used to solicit orders from the investors.
- *Underwriting agreement* - it contains details of the agreement between a company and an underwriter.
- A *client-stockbroker agreement* - it specifies the particulars of their relationships, e.g. the stockbroker's commission rate and payment date.
- A client's stock *order form* - this document contains different information related to the clients's intention to trade on the exchange, as mentioned previously.
- *Confirmation* - a document sent by a stockbroker to a client recording information regarding a purchase of sale such as price of shares and brokerage.
- *Share certificate* - represents an evidence of share ownership and is also an instrument for the transfer of the title.
- *A stock contract* - this is a typical contract which governs sale and purchase of shares and the transfer of title. This is the most common contract in any stock exchange. It defines the obligations of stockholders, buyers and sellers in a share trading transaction.

Note that the contract templates identified are relatively simple, since there is no ongoing relationship associated with the contracts, and the items being traded (shares and currency) are well defined and have minimal complexity. Clearly, the inter-organisational interactions will require more complex contract specification as well as contracts which are required to specify different, technology related QoS details.

Most stock contract templates are composed of a number of common *contract elements*. For example, the stock contract template includes standard contract elements such as:

- the *roles* of the stockbrokers (e.g. buyer and seller)

22. The Companies Act contains a quite extensive schedule of information that must be included in the prospectus [2]. However, these are beyond the scope of this example.

- *money* (as being offered by the buyer), and *shares* (as being offered by the seller); these representing a consideration aspect of a contract
- other obligations, including the requirement the *settlement date* (e.g. that exchange of shares and money take place within 5 days)
- the statement that the contract is immediately effective, viz. immediately after an offer acceptance has been received by the stockbroker who sent the offer.

There may be different types of stock contracts, which are related to different types of stocks that they refer to, for instance, common stock and preferred stock and relationships such as this need to be appropriately supported. These standardised contract templates can be stored within a dedicated *stocks contract repository* (or several repositories, if say security reasons dictate their separation). This function can for example be done by an authorised party on behalf of the ASX. Once such fixed contract templates are stored, they can be instantiated as needed.

In addition to the stocks contract repository, a number of other specific repositories can have a role of storing the pertinent instances, such as the repositories for the instances of share certificates, approved prospectuses and underwriting agreements. These can be included within one or more *notaries*.

Negotiation in the stock exchange is achieved by having the system match offers to buy with offers to sell. An exact price match must be achieved for the successful establishment of a contract to purchase. Offers are placed publicly with an open access for all stockbrokers—the negotiating environment is hence competitive, rather than strictly two-party[23]. In this environment, the contract template is fixed, with negotiation based purely on parameters (i.e. price and number/type of shares).

In relation to the *validity* of contracts in the electronic stock exchange, the following applies.

23. On the contrary, in another type of security market, over-the counter market (OTC), negotiation is performed between two stockbrokers [38].

- Agreement and understanding of contract terms is achieved through system-directed negotiation using fixed templates.
- Checking of the consideration is reduced to the type-checking of the contract element types which represent currency and stocks.
- Competence is implicitly guaranteed by the fact that the stockbrokers have the registered status. However, an extended system can be envisaged that identifies the buyer for whom stocks are being traded, allowing checks for such things as buyer bankruptcy.
- The legal purpose of the share trading contract cannot be fully determined at the time of trading. For example, certain industries have minimum Australian ownership constraints, and purchaser information is not currently available on the system. In this case, manual, retrospective action can be used to rectify any violations of the law. A more complete system that identified the purchaser of shares could avoid such problems automatically.

These operations can be implemented through a *stock contract validity* component.

In the stock exchange, *monitoring* is carried out by the parties themselves, with any violations reported to the ASX. Alternatively, a special component (i.e *stock contract monitor*) can perform a number of these tasks automatically, as for example implemented in the ASX surveillance system [7]. Note that there is no strong requirement to monitor QoS, since there is no scope for variation in the quality offered by stockbrokers in a share trading contract.

The *enforcement* is carried our reactively, with financial penalties for late or non-settlement of the contract. This is carried out automatically once a violation is detected and can be implemented within the corresponding *stock contract enforcer* component. In addition, post-contract deregistration of stockbrokers can take place when regular violations occur, although this is carried out manually.

An example of the use of the BCA for an ODS component: the RM-ODP trader

The fundamental concepts of the RM-ODP trader have been presented in subsection 2.5.2. The concept of a contract which is needed to govern business interactions between the members of the trading community does not explicitly appear in the current version of the trader standard. In this example we will show how the definitions of different types of trader policies and our discussion related to the trader's economic perspective (section 3.11), in particular the notion of the trader ownership, can be used to introduce the concept of a business contract within the scope of the RM-ODP trading service. It will be then shown how our BCA can be applied in this context.

We have presented some possible scenarios associated with the operational environment of the trader (section 3.11). Considering these scenarios it can be concluded that the *contract domain* of the trading service[24] will be influenced by enterprise issues such as the following.

- The *ownership* of the trader. This includes a private ownership or a publicly available trader (as can be offered by some government bodies).
- The *purpose* for which the trading service is established. This purpose can be the trading for general services or some special purpose trading, for example, in telecommunications, where a trader can be used for real time trading, such as the dynamic configuration of services within telecommunication switches [67]).
- The *jurisdiction* to which the trader belongs.

If the trader community is under the authority of an enterprise, then this enterprise will prescribe the interaction rules of the members of the community, as well as the rules and policies of the trader contract domain. This includes a set of *contract validity* rules and the procedures to arbitrate contract disputes and enforce the members' behaviour. Depending on the size of the community and the divergence of the goals of its members these rules will have different levels of complexity and precision. In the case of such a governance

24. The rules and policies of the trader contract domain represent part of the *trading community policy* (this is a superset of all those policies relevant to the trading service; this includes the trader policy as well as importer and exporter policies).

structure we anticipate that all necessary contract types will be stored in a *trader contracts repository* and the role of LRR will be minimised (e.g. to cover the interactions with other trading domains). If, on the other hand, the trading community members are independent business entities interacting within market environment, then the trader contract domain's rules and policies will be to a greater extent constrained by the jurisdiction to which the members of community belong, normally a Corporation Law of a country or a state (or an appropriate international body of law).

Different *contract types* can be used to govern specific relationships between the parties of a trading community and their semantics will be determined by the following:

- characteristics of the members of the trading community
- the competitive environment of the community
- a character of their relationships, i.e. whether it is an ongoing relationship, a 'one-off' type of relationship or some mixture of both.

We anticipate that most of the contractual arrangements associated with the trading service would be based on a small number of common contract types. In addition, other business contracts (which will be applicable to various trader operational environments and which could cover different interactions of the trading community members) can be derived from these base types[25].

The major trading business contract types will be now identified. We note that each of the contract types specifies the roles of the parties, and their obligations. Part of the obligations encompasses the exact scope of trader service to be provided (based on which the related trader's QoS can be derived, as presented in subsection 4.5.2), and the obligations of other members of the community, such as payment details that apply to importers and exporters. These obligations constitute a consideration element of the contract. In addition to these generic contract elements, a complete contract between the members should also include their interaction specific contract details, as given below.

25. For example, by using an ODS type system as will be described in the next section.

- A contract between a *trader* and an *exporter*. The specific elements of this contract reflect a trader's *service offer acceptance policy*. This is the policy of the trader object which restricts the set of service offers that will be accepted by the trader for inclusions in its repository of service offers (e.g. according to the type, cost, expiry date, trustworthiness of service offers). Since the interaction between a trader and an exporter typically exists over a long time period and because it may involve substantial costs (especially on the side of an exporter), some form of contracts between them can be required to minimise business risks. Following the spirit of transaction cost economics, the exact form of the contract will depend on environmental factors, on the characteristics of the trader and on the exporter and the nature of their relationship (as analysed in subsection 3.4.1).
- A contract between a *trader* and an *importer*. This contract can contain a clause related to the trader's *import request acceptance policy*, according to which the trader specifies the types of requests that can be processed within the trader. Depending on the nature of the relationship between these two members, there may be different requirements on the structure of their contracts. We anticipate that in many situations the interaction between these two members would not require an agreement on explicit contracts between them unless there is an ongoing relationship between these two.
- A contract between *two traders*, which covers terms of their interworking. This contract should include the *domain boundary crossing policies* of both traders. These policies guide the trader in crossing boundaries of domains, such as type, domains, security domains and technology domains [11].

As an example, a contract which governs the exporter-trader interaction will include the following contract elements:

- the *roles* of the members (an exporter and a trader)
- the *QoS parameters* of a trader if relevant for an exporter
- the *payment* scheme for the exporter's use of the export aspect of the trader's service
- any other specific details as mentioned above.

Most *negotiation* activities within a trading community will be performed between the community member pairs, such as <trader-exporter>, <trader-importer> and <trader-trader>. The characteristics of the negotiation process (e.g. the duration of the negotiation and the parties' satisfaction with the outcome) will depend on the market competitive forces; these being determined by the number of importers, exporters and trader service providers, as well as quality of trader service offerings.

Monitoring will depend on the governance structure. In the case of a private enterprise, some internal monitoring mechanisms can be implemented, and these can be done mainly by the members of the trading community to reduce the cost of establishing a special agent for this. In a market environment, monitoring can be done by the members themselves or by a special third-party, depending on the priority and importance of the importers. The trader can provide a 'QoS interface' which can be used for monitoring purposes.

Enforcement can be applied to any members of the trading community, depending on the specifics of the underlying contracts. To this end, a mechanism must be provided within each of the members by which the contract enforcer can influence their behaviour.

6.5 Relation of the BCA and a generic ODS architecture (ODS-A)

In this section we discuss how the BCA developed in the previous section can be positioned in relation to a generic ODS architecture. We first consider this in the context of the concepts of an object model and a type system which exist in most of the commercially available ODSs. Following this we discuss the use of the concept of binding for support of the interactions associated with contract operations (subsection 6.5.8.). The discussion in this section will be further expanded in section 6.6, by using a specific ODS environment, the OMG CORBA platform.

6.5.1 The roles of an object model and a type system

Most commercial ODS environments (e.g. ANSAware [3], TINA [9], OMG CORBA

[108], DCE [126]), as well as the RM-ODP [66], [82], adopt an *object based* approach which normally includes the following two components.

1. An *object model*, which is used for the encapsulation of the internal structures of programming entities (both at the application and the infrastructure level) while emphasizing the visible interactions between the entities. These interacting entities are referred to as *objects*; they encapsulate state of objects together with operations defined on that state [34]. Objects are accessed through the *interfaces* which define named *operations* together with constraints on their *invocation* (an object can have one [108] or more interfaces [73], [66]). *Activity* takes place when objects invoke named operations in the interfaces of other objects.
2. A *type system*, whose purpose is to facilitate common understanding and interoperability in an ODS [22]. This typically includes the following two major subsystems.
   a) A *type model*, which can support the description of the structure, behaviour and semantics of *types* (types are usually defined as predicates; for example, the RM-ODP definition is: 'Type is a predicate characterising a collection of <X>s' [65]).

      A type model includes the description of different types: these can range from the fundamental types to the very complex types. An Interface Definition Languages (IDL) is used to describe the types of a type model. The usual types, which are present in most of the IDLs [3], [108], [126] are:

      - *basic data types* (e.g. integers, real numbers, and characters)
      - *basic type constructors* (e.g. records, unions, structures, and sequences)
      - *interface signatures*, which define parameters of operations, including their data types.

      In addition, IDLs can have the notions of:

      - *object types*, which are used to represent the operations provided by the objects and can include their behaviour
      - *interface types*, which support the description of the external behaviour of the objects.

      A type model can also include the definition of more complex types [22], [66], [108]. One such type, a *binding type*, is particularly useful to address the require-

ments of complex services, such as multimedia and multiparty services, and we contend that this can be also beneficial in modelling the interactions associated with business contracts (as will be discussed in subsection 6.5.8).

3. A *type manager* (or a type management system), which provides persistent type information through a type repository. A type repository contains a type hierarchy defining possible relationships between types, including the subtype or substitutability relationships. A type management system also provides operations to query and manipulate types and facilitates the instantiation of the instances of different types from their templates.

In implementing concepts of our BCA, we follow the same object based approach. For example, we assume the existence of an object model as a prerequisite to specify the functionality of the BCA components (in terms of the interfaces of the corresponding objects), and the parties to a contract (as a set of one or more objects that implements the behaviour of those parties, both within and beyond the contract). In addition, we assume the presence of a type system for the representation of the notion of contract type, the dependencies between different contract types, and their instantiation and manipulation. We now elaborate how each of the BCA components identified in section 6.4 can be related to the concepts of a generic ODS-A.

6.5.2 Contract repository

We propose that the contract repository (CR) component be incorporated within a *type repository* (Fig. 6.7). The CR can include some general contract elements such as the names of the parties to the contract, consideration elements of the contract, duration of the contract and date of the contract agreement (these are symbolically represented as *party_A, party_B, party_A_gives* and so on). These contract elements can be defined by using basic data types (e.g. integers, characters).

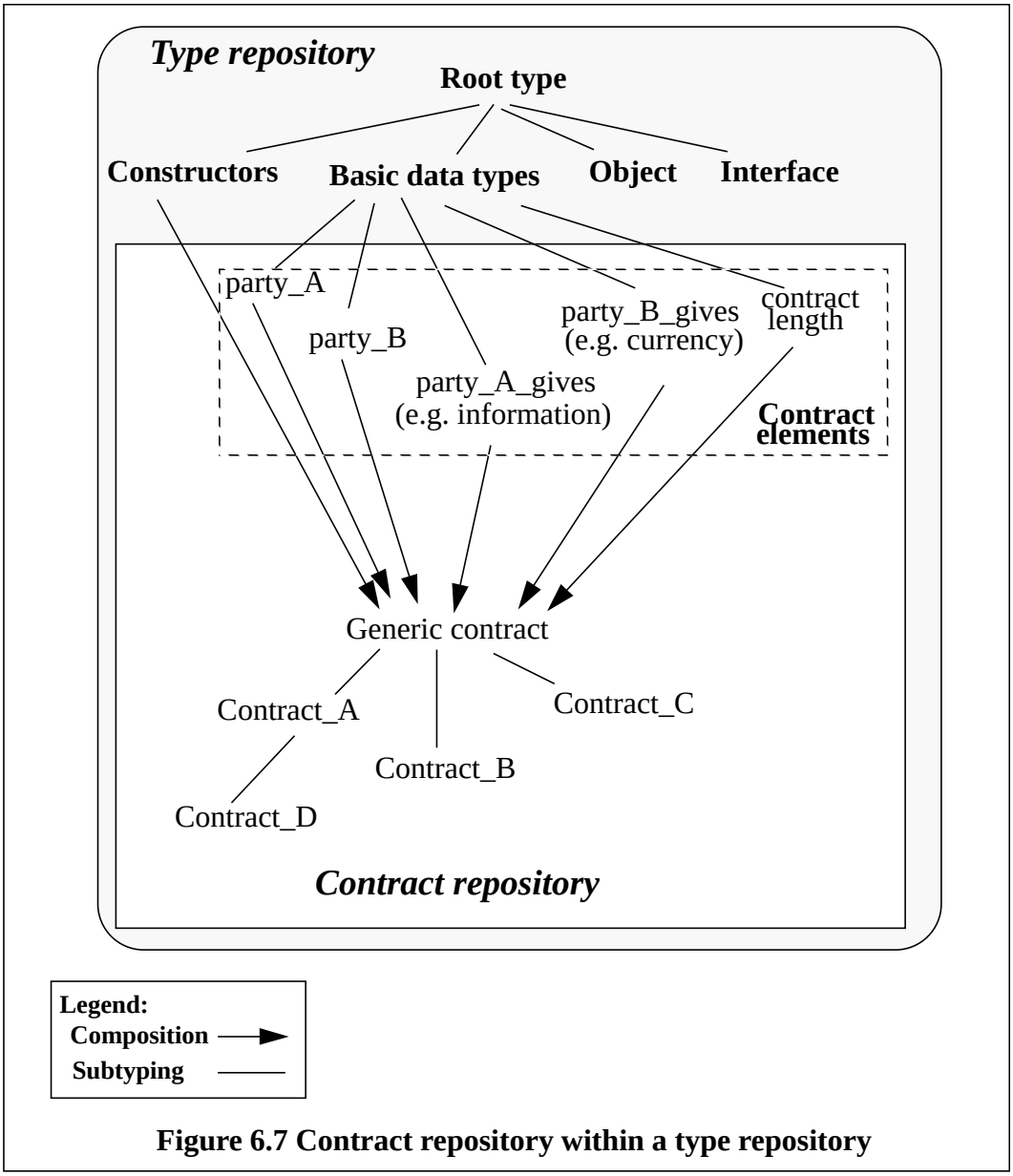

Figure 6.7 Contract repository within a type repository

Further, a generic contract type can be constructed by using the contract elements and an appropriate type constructor. For example, the generic contract type can be composed of typical contract elements by using a *structure* type. This generic contract type can then be used to derive special contract types. Examples are a real-estate contract for house purchase and various insurance contracts (the derived contract types are annotated in Fig. 6.7 as *contract_A, contract_B* and so on). The subtype relationships between the contract types can be represented through the contract type hierarchy, as depicted in Fig. 6.7.

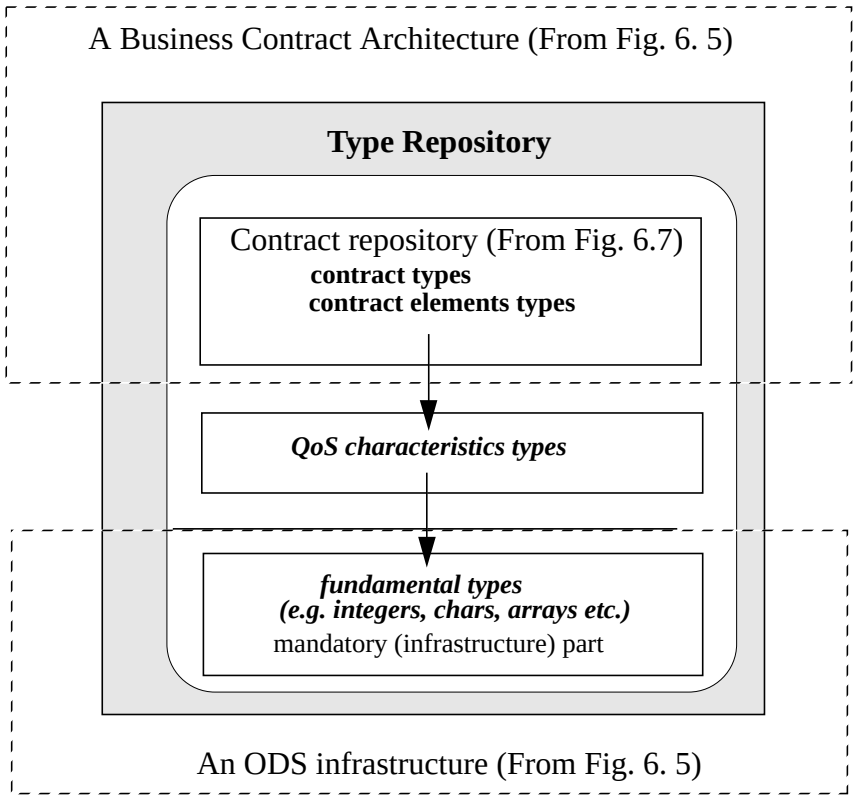

Figure 6.8 Type Manager: a link between the BCA and ODS infrastructure

Figure 6.8 highlights the fact that a type system plays a significant role as a link between the BCA and an ODS infrastructure. In this figure we have also introduced 'QoS characteristics' types which can be used for the construction of those contract types which require a representation of QoS characteristics (of multimedia services say). In other words, in addition to the types stored in the CR, a type repository can also include a collection of technology-related QoS characteristics types such as delay, availability, reliability and security (as identified in section 4.3). These types can be used by the contractual part of the type system (but also by other architectural components and by other functions of an ODS, if needed). They could be employed to represent physical characteristics of underlying resources and can be exploited for different binding phases such as QoS negotiation and the connection establishment. The architectural notion of QoS is discussed in more detail in subsection 6.5.8 and section 6.8.

6.5.3 Contract domain

In terms of an ODS architecture, a business contract domain can be defined as a scope or a boundary within which the *activities of objects* are regulated by the jurisdiction of that domain. The jurisdiction administers the rules and policies associated with a particular legislative authority and rules which govern interactions with other domains. These rules and policies, stored in the legal rules repository, should be made accessible via the CL component [101]. This component could be implemented as a single object or a set of co-operating objects.

A contract domain can be realised in a variety of ways such as: *i)* a scenario in which the domain's jurisdiction completely and solely controls a number of nodes and the networks between them, or *ii)* a situation in which parties and contracts within the domain are distributed as objects and bindings throughout an ODS in which the authority of the jurisdiction over its members is retained.

Due to the similarity of many aspects of management and business contract issues in distributed systems, the paradigm of domains developed for management purposes in [104], [134], can be adapted to further refine business contract domain operations and relationships between domains. This similarity will be investigated in future work. In particular, it is worth noting that there may be different relations between domains, such as a hierarchical or peer-to-peer relationships [101].

6.5.4 Contract negotiation

In the context of an ODS-A, contract negotiation can be realised as the actualisation of parameters and the refinement of contract templates (by the selection of subtypes). This reflects the fact that negotiation often involves one party which submits a contract offer, which other party should either accept immediately or make a counter-offer. The subtype hierarchy can be used to determine the substitutability of contract templates. Substitutability can be used as the basis for acceptance of counter-offers [101].

6.5.5 Contract validation

Contract validation is an optional step performed individually or collectively by parties to the contract. As described in subsection 6.3.3, there are four elements of a legally valid contract. The realisation of these in elements in terms of the types and objects of an ODS can be as follows [101]:

- *An agreement.* The existence of an agreement is proven by the record of negotiation kept by a notary object. Clarity is implied by a common understanding of the types associated with the contract. This can be realised by using an ODS type system.
- *Consideration*. The existence of consideration is ensured by requiring that contract templates include a description of what is exchanged by the parties to the contract. The jurisdiction prescribes what exchanges are considered acceptable consideration through definition of types in the type system. We envisage that the most common forms of consideration in an ODS type model will be information and currency.
- *Competence*. Competence is assessed through certification authorities that can assess the competence of parties to fulfil particular roles, e.g. a credit rating agency can assess the ability of a company to meet financial obligations of a contract. Typically parties demonstrate their competence to such certification authorities in advance to streamline the checking of their competence when establishing contracts. We do not constrain the establishment or operation of these authorities. Since in a general case this checking involves some form of authentication of parties, this would require an agreement for the established procedure for this. We assume that such a security procedure is established and concentrate on contract specific issues.
- *Legal purpose*. Legal purpose can only be determined through interpretation of the laws of the contract domain stored in the legal rules repository, e.g. a business law repository. The legal purpose constrains the actions that may be performed by the objects representing the parties to the contract. Such constraints can be implemented by requiring that all contract templates satisfy constraints specified as types of the type management system. Formulation of the types is a responsibility of the jurisdiction. However, the interpretation of law requires either human intervention or an artificial intelligence approach such as that suggested in [81].

To summarise, contract validation can be realised by using rules and policies of the underlying business contract domain and through implementation of appropriate supporting services to facilitate validity checking.

6.5.6 Contract monitoring

During the existence of a contractual arrangement either object may check whether the contractual obligations have been met during the contract realisation. This requires recording of the actions and measurement of performance of parties, ensuring that they comply with the contract specification. This process of contract monitoring can be done by the objects themselves (and possibly verified later if needed; similar to maintaining diaries which can be regarded as some sort of legal documents). Alternatively, a trusted third-party Contract Monitor (CM) object can be used. The parties employing the CM object specify the actions required upon detection of a contract non-performance, e.g. it can notify an appropriate contract enforcement object.

6.5.7 Contract enforcement

Contract *enforcement* can be done via third party objects which can:

- take proactive corrective actions to minimise the deviation from the contract, during the execution of the contract terms agreed previously
- constrain future activities within the contract domain for parties that have violated contracts.

6.5.8 Contract types and binding types

The concept of a binding is introduced to represent the idea of connecting objects together so that they can interact [22]. Most existing distributed programming environments (e.g. ANSAware [3], CORBA [108] and DCE [126]) have a limited view of binding: a connec-

tion between two objects. More general concepts of binding have also been proposed. These allow for multiple parties and, to varying degrees, allow the different roles of the parties to be specified. For example, TINA has introduced such a concept in response to the need to provide a mechanism for modelling complex multimedia services. It is able to support many communication configurations which include multiple participants and which require more complex control. The complex system constituting this binding object is referred to as the Connection Management Architecture [53].

A general concept of binding is defined in the RM-ODP, i.e: 'binding object template consists of:

- a set of formal role parameters
- a set of interface signatures, corresponding to the formal role parameters
- a set of control interface templates
- the behaviour to occur when the template is instantiated
- an environment contract[26].'

A similar notion of binding is also proposed in [16]. It includes multiple participants, a definition of the flow of information between participants and a specification of obligations and requirements of each of the participants and of the environment itself [22]. Such a binding has a *type* which specifies the binding *semantics* and the *roles* that can or must exist in binding. Binding semantics provides a template for interactions between objects. Roles of a binding are filled by the objects participating in the binding. Each role of a binding specifies an interface type that must be satisfied by objects fulfilling that role.

We believe that the concept of a binding type can be used as a suitable means to model enterprise related interactions, such as those associated with business contracts. As discussed in section 6.4 there may a large (but still finite) number of scenarios associated with business contract interactions. These for example can be related to different:

- temporal ordering of interactions

26. An environment contract is 'a contract between a computational object and its environment, including QoS constraints, usage and management constraints [66]'.

- dependencies between the entities to the contract
- dynamically changing number of participants involved in the contract operations during the contract life time (both the parties to the contract and the BCA components)
- relationships between contract domains.

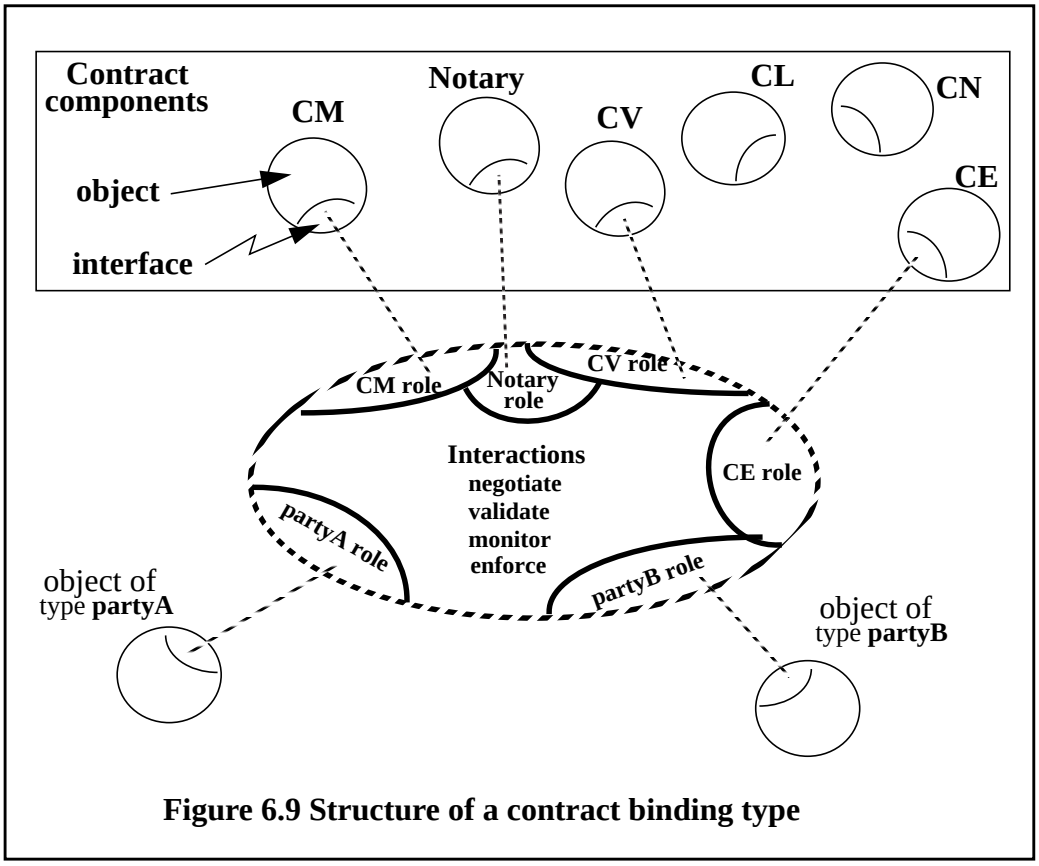

Figure 6.9 Structure of a contract binding type

We contend that different contractual scenarios can be described through the corresponding binding types. A binding type can specify the maximum number of the participants to the contract, the corresponding roles, and also the possible interactions that can be supported within that binding type. An example of such a binding type, referred to as a 'contract binding type' hereafter, is depicted in Fig. 6.9. This binding type:

- includes the roles of *CV, Notary, CM* and *CE* (but not *CN* or *CL*) and the roles of two parties to the contract, *partyA* and *partyB* (in general a binding can specify roles of many parties to the contract)

- supports interactions which embody contract validation, notary operations, contract monitoring and contract enforcement.

Since various binding types correspond to different contractual arrangements, we anticipate various relationships between them. For example, the *subtyping* relationships can be used to form an appropriate 'contract bindings type hierarchy', in which a base 'contract binding type' can be a subtype derived from the binding type (Fig. 6.10).

In addition, we anticipate that frequently used contractual interactions can be embodied in the corresponding common contract binding types and these can then be used to construct required contract binding types. As an example of one such contract binding type we introduce a 'contract binding negotiation type'. This contract binding type can describe basic negotiation related interactions associated with contract templates i.e., offer, counter-offer, acceptance and rejection. In the case of a successful negotiation, this binding type specifies that these interactions are recorded by a notary object for use as evidence of agreement in the contract validation and enforcement activities. Another example is a 'contract enforcer binding type'. This could provide support for contract enforcement and it could be done either *i)* through constraints implemented within the binding between the representative objects (this can potentially be implemented through dynamic type checking of interactions between objects) or *ii)* via third party objects which might or might not be part of the binding, taking corrective actions to minimise the deviation from the contract.

We now draw the correspondences between this proposal and the approach used in the previous subsection. The contract binding types correspond to the contract types discussed previously. For example, the 'generic contract binding type' corresponds to the notion of generic contract mentioned in subsection 6.5.2. The roles of contract binding types correspond to different contract elements. Finally, there is no notion of interaction within contract templates in the previous subsections (as the corresponding contract types are effectively data types and they do not embody an activity).

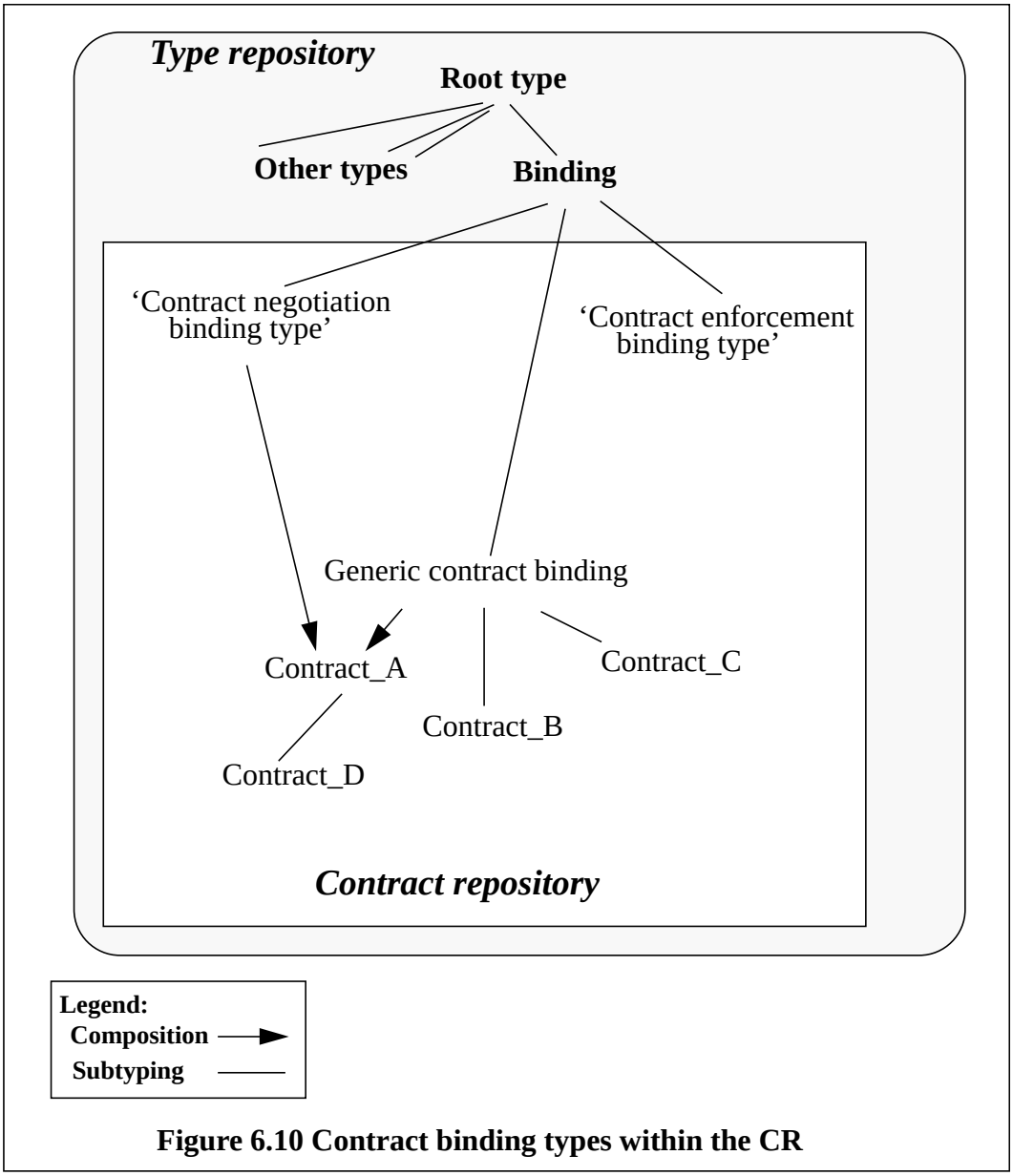

Figure 6.10 Contract binding types within the CR

We also note that the use of binding type to support enterprise related interactions is in conformance to the RM-ODP, where binding is defined as 'a contractual context, resulting from a given establishing behaviour' [65].

The suitability of the use of binding type is further augmented by the fact that in addition to being used as part of the business contract modelling, this type can be used to represent any non-functional aspects of interactions such as *QoS* (e.g. time constraints on interactions, required bandwidth and reliability). In fact, we anticipate that this (or similar) types

will be introduced in ODS type models, so that the computational modelling requirements of future complex services which involve multi-party interactions and multiple media types can be met (as also suggested in [53]).

In view of this integration of business contract interactions and computational QoS details, we emphasize that it is the combination of the business specific contract types and technology-related QoS types that should be satisfied during the binding process, if it is to successfully complete. To illustrate this, we depict a general binding process within an ODS-A (Fig.6.11).

This picture represents a complex binding process, which involves the fundamental infrastructure components such as type manager, and also a number of optional architectural components such as various security components (e.g. for authentication, authorisation and encryption), the trader and a set of the BCA components identified in section 6.4. Note that the 'binder' object symbolically represents that part of the infrastructure which performs the binding function. In this example there are three parties to the contract and they are represented with the objects O_1, O_2 and O_3, while the binding template specifies three roles viz R1, R2 and R3 (the roles of the BCA components are not shown for the simplicity of illustration).

As indicated previously, the type manager contains a number of mandatory types, and also the types which describe different QoS characteristics, including those that refer to technology related variables. The optional architectural components will be accessed during the binding process in order to resolve various application-specific or system-specific requirements. For example, the relevant contract components will be accessed to ensure establishment of legally valid contracts between the objects in the system. Hence, this binding will successfully complete (with the creation of a *binding instance*) if the following conditions are satisfied:

- the objects' interfaces types are matched with the roles of the binding
- the contractual operations are successfully performed (e.g through different contractual components)

- and the low-level QoS requirements are met.

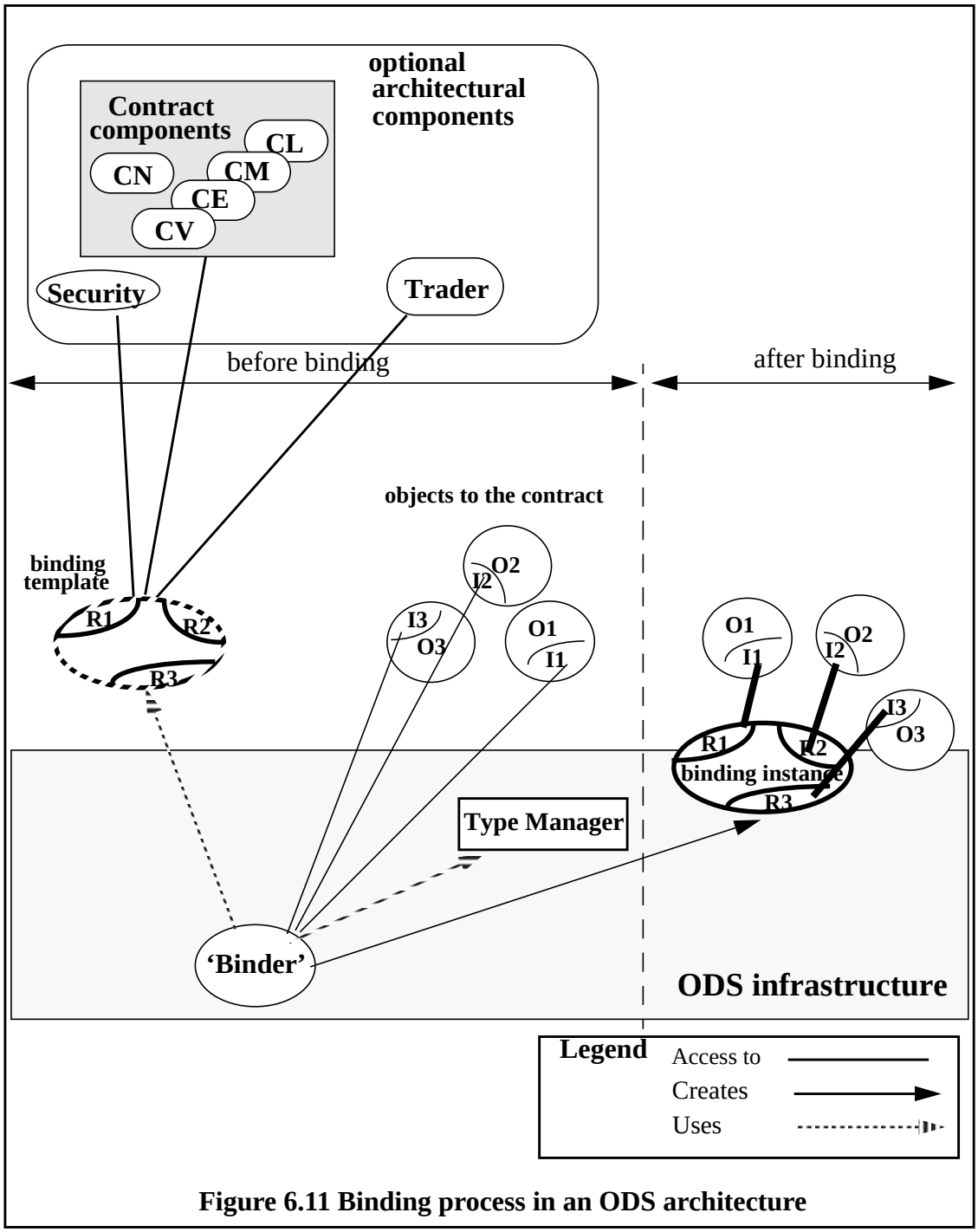

Figure 6.11 Binding process in an ODS architecture

6.6 Relation of the BCA and a specific ODS-A: the CORBA model

In this section we endeavour to map the BCA proposed onto a specific ODS infrastruc-

ture, the OMG's Common Object Request Broker Architecture (CORBA) model. We will first outline the OMG's Object Management Architecture (OMA) Reference Model, and then the CORBA specification, which represents the OMG adopted specification of the central part of the OMA, the Object Request Broker[27]. The two illustrative example are also provided as before, the electronic stock exchange and a trader service, followed by the discussion of the use of CORBA concepts for our BCA.

6.6.1 Outline of the Object Management Architecture Reference Model

OMG's mission is to 'develop a single architecture, using object technology, for distributed application integration, guaranteeing: reusability of components, interoperability and portability and basis in commercially available software' [136]. The OMG's view of accomplishing this is by having a consensus based approach for the definition of:

1. a single terminology for object orientation
2. a common abstract framework
3. a common reference model
4. the common interfaces and protocols.

The Object Management Architecture Reference Model (Fig. 6.12) classifies the components, interfaces and protocols which compose an object model into four areas [108].

- The *Object Request Broker* enables objects to make and receive requests and responses in a distributed environment.
- *Object Services* is a collection of fundamental services (interfaces and objects) that provide basic functions for using and implementing objects.
- *Common Facilities* is a collection of higher level services broadly applicable to many applications of high value capabilities or vertical markets.
- *Application Objects* are objects specific to particular products or end-user systems.

27. We note that the generic term Open Distributed System Architecture (ODS-A) used throughout this thesis corresponds to the OMA Reference Model, while the term ODS infrastructure corresponds to CORBA.

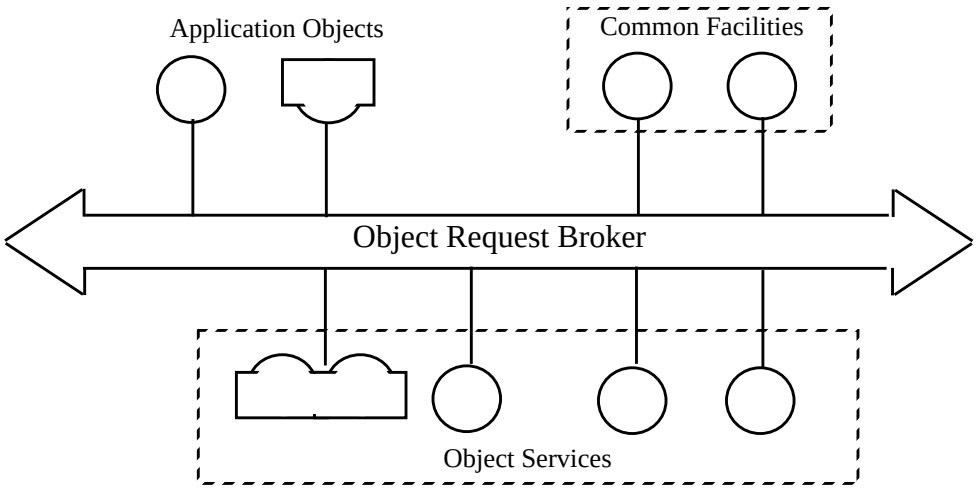

Figure 6.12 Object Management Architecture Reference Model

The Object Request Broker (ORB) provides the basic communication channel through which objects interact to provide system services. The ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems [108]. In fact, the ORB was the initial focus of OMG effort. This has been followed by the effort on identifying object services, common facilities and application objects: an evolving process of adopting new technologies and populating and enriching the core of ORB facilities.

The structure of the OMA can be metaphorically explained as: *i)* the communication protocol defined by the ORB being the grammar of all other OMA specifications (since all object behaviour is defined in terms of messages exchanged among the objects) [111], *ii)* Object Services representing a system's most basic vocabulary (the essential services needed to create an object, introduce it into its environment, use and modify its features) [111], *iii)* Common Facilities being a common and shared terminology to be used by different application areas and *iv)* Application Objects, being specific terminology pertinent to particular types of end-user systems or commercial products.

Examples of the concepts and services provided by different OMA components are as fol-

lows.

- An ORB architecture includes fundamental concepts such as the underlying object model, and the structure of ORB. The Common ORB Architecture (CORBA) represents the adopted ORB technology by the OMG [109] and it will be discussed in more detail below.
- The Object Services include services such as naming, event notification, lifecycle services, transactions and relationships. Future object services include security, time and a trading service (based on the RM-ODP trader).
- Common Facilities encompass [111]:

  a) The *horizontal* set of common facilities, which are shared by many or most of the systems, regardless of application content area. Four major domain of facilities identified so far are: user interface, information management, systems management and task management.

  b) The *vertical* market facilities represent technology that supports various vertical market segments such as health care, retailing and financial systems.

  We note that the BCA developed in this thesis can be positioned within the horizontal set of common facilities, since it is based on a *generic* contractual framework (the BCF). It can then be used as part of the vertical market facilities such as accounting facility and Information Superhighways facility [111].

- Application objects include *business objects.* These objects describe real world objects, which are of crucial importance for functioning of businesses, for example customers, products, financial instruments and orders. The business objects denote an important set of concepts that are expected to represent the major thrust of software industry in years to come [128]. The process of defining business objects and populating the application objects area with them is currently undergoing within the OMG's Business Object Management Special Interest Group[28].

28. This is one of the several Special Interest Groups (SIGs) within OMG. Examples of other groups include end-user requirements, telecommunications and healthcare [136].

6.6.2 The CORBA object model and type system

In this subsection we outline the fundamental concepts of the CORBA architecture and specification, as adopted by OMG[29]. This will be in terms of its object model and type system (as introduced in subsection 6.5.1).

The CORBA *object model* is centered around the following concepts [109].

- An *object* is an identifiable, encapsulated entity that provides one or more services that can be requested by a client. An *object reference* is an object name that reliably denotes a particular object.
- Client request services by issuing *requests*. A request is an event, i.e. something that occurs at a particular time; it causes a service to be performed on behalf of the client. The information associated with a request includes an operation, a target object and zero or more (actual) parameters. An *operation* is an identifiable entity that denotes a service that can be requested; it has a signature that describes legitimate values of request parameters and returns results.
- An *interface* is a description of a set of possible operations that a client may request of an object. An interface may have one or more *attributes*. An attribute is an identifiable association between an object and a value. It is logically equivalent to declaring a pair of accessor functions: one to retrieve the value of the attribute and one to set the value of attribute. An attribute may be read-only, in which case only the retrieval function is defined. We note that each CORBA object provides a single interface (this is different to other models, such as the RM-ODP or TINA, which permit an object to have multiple interfaces).
- An *object implementation* is a definition that provides the information needed to create an object and to allow the object to participate in providing an appropriate service. An object implementation typically includes definitions of the *methods* that

29. The first version of CORBA (1.0) was adopted by OMG in October 1991. CORBA 2.0 specification was adopted in December 1994; it standardises interoperability between machines running different Object Request Brokers. CORBA 2.0-conformant systems are now being developed and are expected to be released and adopted by independent software developers and end-users by the end of 1995.

operate upon the state of an object (a method is a code that is executed to perform a service). It typically includes information about intended type of the object.

The following concepts of the CORBA *type system*, which are of relevance for the BCA are summarised below.

- A *type* is an identifiable entity with an associated predicate defined over values. A value satisfies a type if the predicate is true for that value. A value that satisfies a type is called a member of the type. For example, an object type is a type whose members are objects. The present CORBA *type model* supports basic types, constructed types, interface types and object[30] types [109]. It does not include a binding type.
- The CORBA *Interface Definition Language (IDL)* is used to specify particular distributed application. The CORBA IDL is an important part of the CORBA specification: it promotes interoperability, reuse, and openness by providing a means to specify interfaces which in turn can have different implementations.
- The CORBA *Interface Repository (IR)* component corresponds to the type repository discussed in subsection 6.5.1. This component provides a persistent storage of type definitions in a CORBA environment. Given an object reference and the object's type, all information about that type can be determined at runtime by calling functions defined by the IR.
- The CORBA *Implementation Repository* keeps a record of implementation details necessary for the ORB to instantiate objects.

6.6.3 BCA and CORBA

The CORBA concepts outlined in the previous subsection can be used to implement our BCA. In this subsection we will summarise main points of how this can be done (in chapter 7 this will be extended with the presentation of a prototype which implements each of the BCA components via a CORBA conformant platform). We emphasise the following important points.

30. We note that in the present OMA model an object has one interface and the notion of object type in this model is not as rich as the RM-ODP object type.

- The CORBA Interface Repository (IR) represents an elementary type repository (though with less functionality than a general type management system, such as the one proposed in [22], or as part of the RM-ODP). The IR thus represents a natural choice for the persistent storage which contains the contract repository with the contract types.
- We argue that an IDL file which contains the description of different contract types can be compiled and stored within the IR by an authorised administrative body, responsible for creating and administering contract templates which are legally approved by relevant jurisdictions. These can then be retrieved at runtime, by both servers and clients, using IR supplied functions.
- Since there is presently no concept which corresponds to a binding type within the CORBA model, the contract type template and contract type hierarchy can be implement by using the CORBA interface facility and operations associated with it. One possible way of accomplishing this is presented in subsection 7.2.2.

6.6.4 Examples of the use of CORBA

In the examples that follow we will illustrate how the relevant concepts of the CORBA model can be used to model the electronic stock exchange application and a trader service, presented in the previous subsection.

The electronic stock exchange example

The semantics of the different kinds of contracts used to facilitate trade in a stock exchange can be represented by the corresponding *stock contract types*. Several specific stock exchange contract types are identified in the previous section[31]. Since these contract types have different purpose, there is no relationship between them. However, there are some specialisation types of relationships, such as the following specialisation from a ge-

31. A more general classification of contracts was given in subsection 6.3.1.

neric contract type towards common stocks contract type: *generic contract -> security contract-> generic stock contract -> common stocks contract -> stock contract*. The stock contracts type hierarchy can represent part of the contract repository, as depicted in Fig. 6.13. We reiterate that due to the lack of a binding type in CORBA, a contract template and contract relationships can be implemented by using the CORBA interface concept.

CORBA objects can be used to model the behaviour of the parties to the stock trade. Each object has one *interface type*, which in turn includes one or more operations. These are annotated in Fig. 6.13. For example, the *client object* and the *company object* should be capable of requesting services of the *stock exchange objects and* the *stockbroker objects*. The stock exchange objects should be able to issue shares, and provide functionality of buying and selling shares (operations A, B and C). The stockbroker objects should be able to perform operations associated with negotiations and the transfer of the title, as described in the previous section (operations D to J).

The main concepts used in this example are depicted in Fig. 6.13.

Trading service

In the previous section we have outlined how our BCA can be used to provide a support for business contracts within the trading community of the RM-ODP. Future versions of the CORBA will also include a trading service and it will be influenced by the RM-ODP trader specification [136].

The notions of a type, an interface and an object, can be exploited in a similar manner as is presented in the previous example. In fact, the specification of trading interfaces (the interfaces defined within the RM-ODP trading function computational specification viz trader, management and interworking interfaces) is already provided using the CORBA IDL (Annex D of the current trader document [67]).

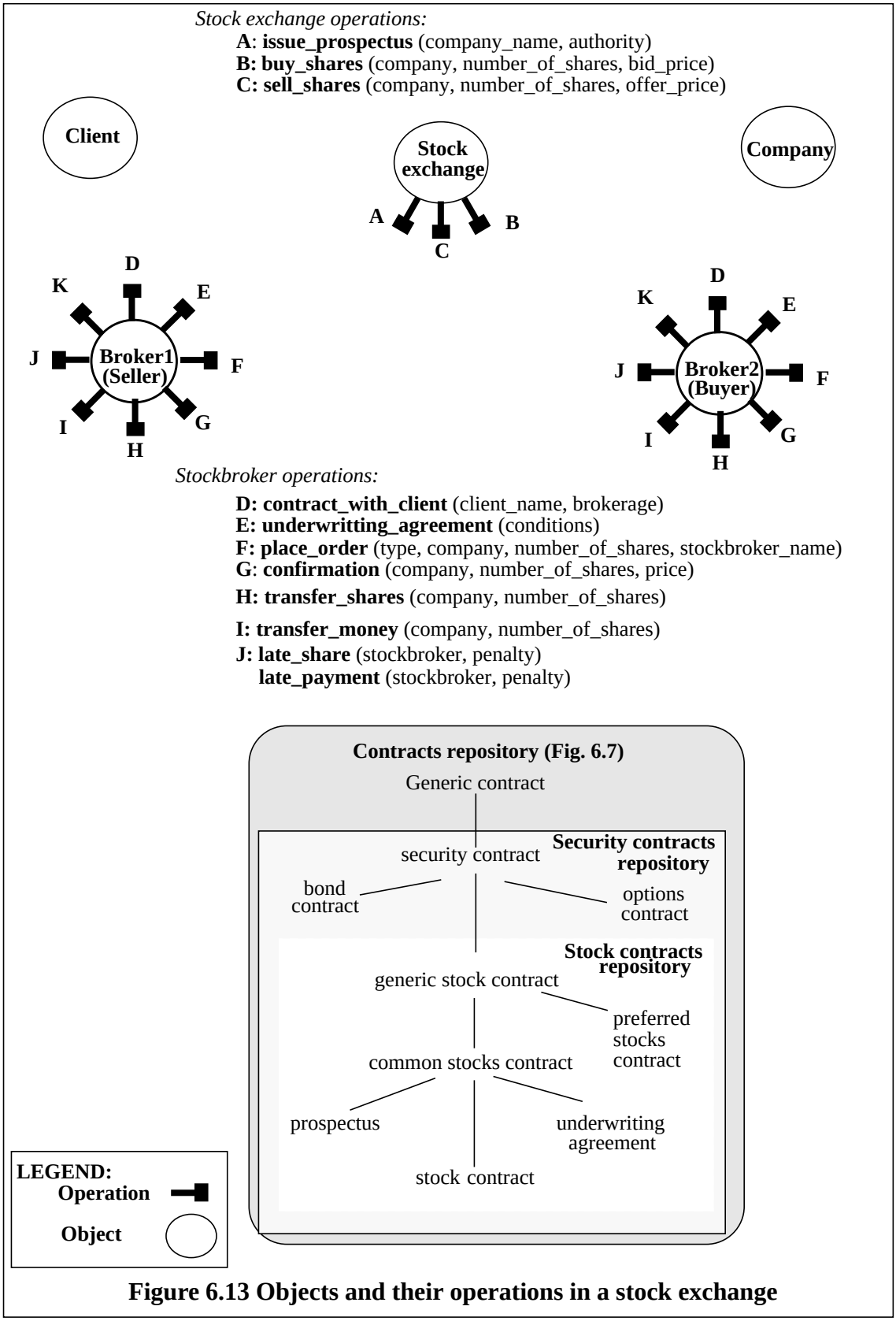

Figure 6.13 Objects and their operations in a stock exchange

In addition, for the modelling of the contractual aspects of a trading service one can also

employ the concepts of our BCA. For example, this can include the interfaces which can be introduced to support interactions relevant for business contract operations such as those termed ‘QoS interface’ (for monitoring purposes) and ‘enforcing interface’), as well as the BCA components identified before[32].

6.7 Contracts and new service requirements in ODSs: the role of QoS

In previous sections, our focus was on an architectural support for business contracts. We used examples of the electronic stock exchange and the RM-ODP trader to illustrate business related issues: how our BCA can be used to accomplish the functions of contract validation, monitoring and enforcing in an ODS. Our BCA is general enough, however, for the various technology-related QoS issues to be specified as a part of contract templates. This is particularly important in view of new service requirements such as distributed multimedia services. In the context of these services QoS can be regarded as a link between the enterprise contract specification and underlying computing and communication resources[33]. When a QoS specification is to be included within the contract specification, the QoS details need to be related to the required and available resource requirements of the ‘middleware’ and the underlying computing and communications resources. To this end, an architectural notion of QoS has to be introduced. In contrast to the stock exchange example, where the characteristics of contracts were such that there were no technology-related notions of QoS, this section introduces the example of a video service, through which it will be possible to discuss different issues related to QoS. We will first illustrate how the QoS (as part of contract specification) can be positioned within a generic ODS-A model and then discuss possible directions for CORBA extensions to support ODS QoS requirements (as identified in section 4.3).

32. We postpone the description of a realisation of these components to chapter 7.
33. Within the context of the RM-ODP, QoS is a part of the environment contract, where ‘environmental contract is a contract between a computational object and its environment, including QoS constraints, usage and management constraints’ [66].

6.7.1 Computational requirements of new services in an ODS

The requirement for supporting distributed multimedia services as an integral part of an ODS infrastructure, brings a need for appropriate architectural mechanisms which would support the physical characteristics of new media such as video and audio. This includes the operations which would support selection of communication ports (as sources and sinks of the flow of multimedia information), connection establishment between these ports, synchronisation between different media flows and the control of individual flows.

Inherent in these new media is the notion of continuous flow of information from its source to its sink. Such a concept cannot adequately be supported by using the invocation model of communication (such as RPC mechanism). There are problems with both possible approaches in an attempt to apply this model to continuous media flow, i.e. to represent continuous media as a result of a single invocation, or to represent it as a sequence of invocations [34]. In the former case, the potentially unbounded nature of continuous media (e.g. continuously running surveillance camera) makes it prohibitive to use a single invocation, while in the latter case there is no way (in ODP computational model) to specify QoS constraints that apply to a sequence of invocations. To address these requirements the new computational abstraction, the *stream* interface, has been proposed (in ANSA [106], TINA [53] and the RM-ODP [66]).

Further, multimedia services require computational support for the representation of relationships between sources and sinks of multimedia information (e.g. multi-party communication session) and dynamic manipulation of the flows (e.g. adding new stream interface). The computational concept of binding as introduced in [53] and [66] is an appropriate mechanism for the modelling of these relationships.

In order to capture both concepts, the traditional *operations* and *streams*, a concept of *interaction* is introduced in the RM-ODP [66]. This is defined as a subset of actions associated with an object which takes place with the participation of the environment of the object[34] [65]. It is worth pointing out that the concept of interaction is suitable not only to

34. A complementary subset of the actions comprises internal actions, which always take place without the participation of the environment.

model the above two types of interaction; it is generic enough to model any type of relationship. This fact was an impetus for the incorporation of a concept of interaction in to the concept of binding in a specific ODS-A [16]. In this model, the binding is defined as a relationship between a set of objects that defines the interaction that can occur amongst the objects during the relationship, and interaction is a ‘visible behaviour that occurs during that association’.

These requirements inherent to new types of media to be manipulated over an ODS are currently being investigated within various organisations. However, to the best of our knowledge, there is no commercially available ODS programming environment which could support these requirements. We note that in current version of CORBA there is no explicit support for these service types.

The telecommunications organisations and groups are perhaps the most active in this domain. For example, the TINA consortium [10] is developing its ODS-A version, known as Distributed Programming Environment (DPE) with the aim of supporting a wide range of broadband applications. TINA includes the concepts of streams and bindings[35]. However, it does not state in detail how to specify, or realise a stream interface as noted in [53].

6.7.2 The role of the binding

It is our contention that a concept of *binding* as reiterated in this subsection (and described in more detail in subsection 6.5.8) holds great promise for encompassing a business contract specification which in turn can contain QoS description. As already pointed out, the importance of positioning QoS representation within a contract specification arises from the interplay of new technological requirements such as distributed multimedia services, and competitive driving factors, which are both embodied in the concept of QoS (as has been highlighted at several points in this thesis). We now turn to the illustration of how the binding can be used to include the concepts discussed in this subsection.

35. TINA has been influenced many relevant standards including the RM-ODP, and makes use of commercially available ODS infrastructures such as CORBA.

In Fig. 6.14, we depict a scenario which illustrates this concept of binding. This figure also depicts those components in which relevant QoS specification elements can be included (in the diagrams in this section the templates are represented with dashed lines and instances with solid lines). This figure presents a case of simple binding, in that it involves two parties, represented by Object1 and Object2 (e.g. a consumer and a producer). In general, a binding can specify several interactions between multiple parties. For example, a binding that implements a house purchase transaction would need to be capable of supporting interactions between a purchaser, house builder (or a real estate agent) and bank.

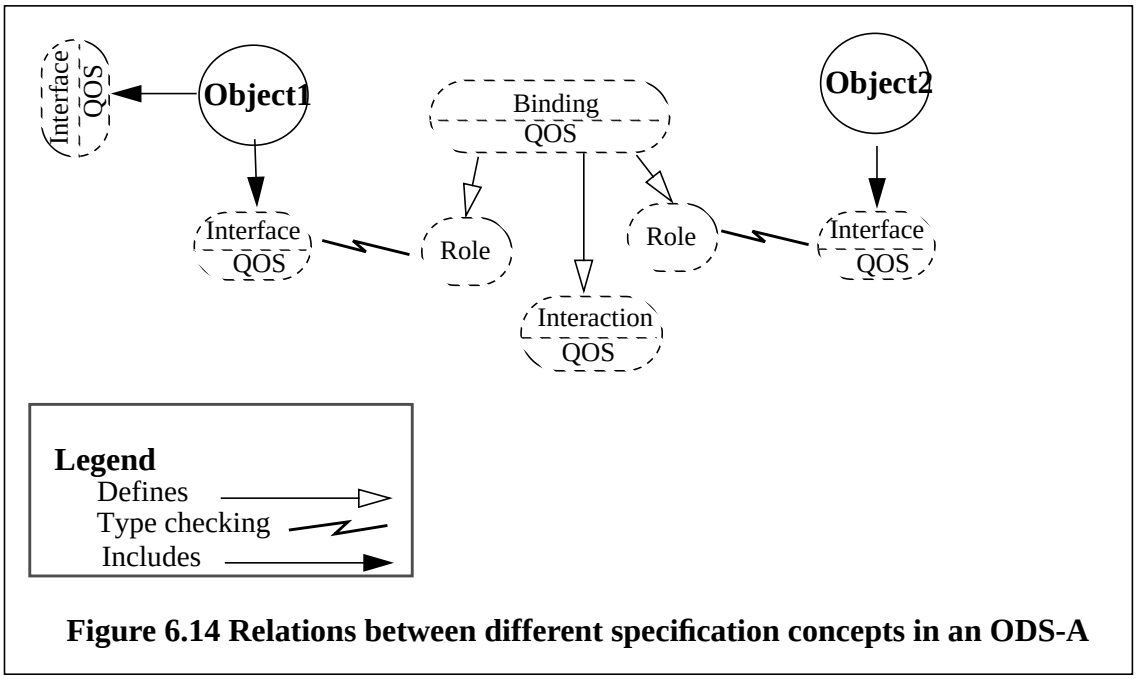

Figure 6.14 Relations between different specification concepts in an ODS-A

The concepts mentioned above will be illustrated using an example of a (two-party) video service type.

Example of an application: video service type

We start with the specification of this video (Fig. 6.15), as follows.

In this example, a User object requires a video stream with a minimum frame rate of (say) 25 frames/s (a video stream is generally characterised with specific QoS characteristics

such as range of values for latency, frame rate, jitter etc.; for simplicity we will focus on frame rate only).

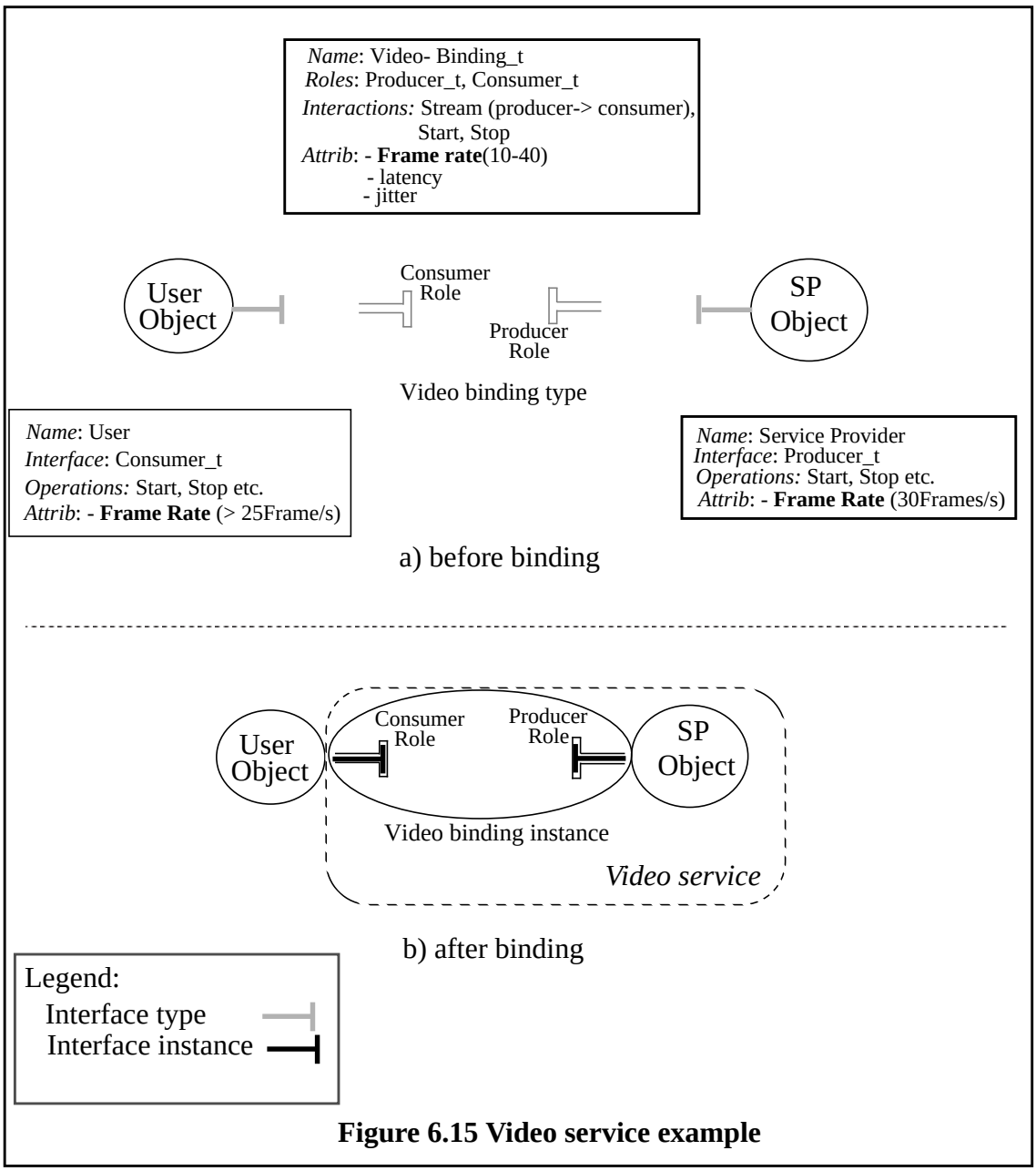

Figure 6.15 Video service example

Following the previous terminology, this video service can be offered to a User object (which has a consumer role) by a video Service Provider (SP) object (which has a producer role) via the corresponding video binding type. Assume that a particular SP object offers its capability to produce video streams with 30 frames/s, as expressed in the specification of its video interface type (this pre-binding situation is depicted in Fig. 6.15a). Assume also that the two party video binding type supports video interactions in

the range of 10-40 frames/s, so that it would accept all SPs and users within this range. This binding type also specifies roles (e.g a SP as a producer of a video stream with its QoS capabilities and a User, as a consumer, with its own QoS capabilities). It is worthwhile noting that this binding type can include additional business contract related elements such as price, contract expiry date and likes (these are not presented here for the simplification of illustration).

There can be several scenarios for establishing of a video binding between the User object and a SP object. We consider the following two scenarios:

a) The User object has knowledge about the binding type which can support a frame rate of 10-40 frames/s. The User wants to become a consumer and submits to the trader a query to search for an appropriate SP. In other words, the User object then trades for an appropriate service. Upon the trader returning an object identifier of the selected SP, the User will ask the ODS infrastructure to create a two party binding with the SP. Namely, the binding process will proceed by the User issuing a request to the infrastructure to instantiate the corresponding binding instance. This step involves a number of infrastructure actions which, if successfully performed, will represent a completion of the binding process. The infrastructure either creates the binding or not, based on availability of resources to satisfy the User's QoS frame rate requirements.

b) Alternatively, the User object may not have knowledge about available binding types and submits a query to the trader for a suitable SP as in the previous case. In this case, the trader will return an appropriate 2-part binding type as well as an interface reference of a producer that asserts to be able to produce 25 frames/s. Upon this, the binding process proceeds as before.

Figure 6.15b depicts a scenario after the binding process is successfully completed, i.e. the *instances* of the corresponding templates.

It is worth noting that in the case of this service type, the binding type will describe a video interaction type only. In general, if a more complex service is specified (e.g a Tourist Information Service type, as described in Chapter 4), then the corresponding binding would

need to describe several additional interactions (e.g billing). This video service has been chosen to simplify the illustration.

6.7.3 (The lack of) QoS support in current CORBA

The current version of CORBA[36] does not support the requirements of distributed multimedia services. For example, there is no support for the concept of stream, nor a notion of a binding type. Clearly, this limits an extent to which multimedia applications can be built using CORBA.

This status of the CORBA development can perhaps be attributed to the absence of strong commercial forces at present to support distributed multimedia over ODS platforms such as CORBA, and the lack of commercial experience with such services[37]. Since applications represent a major driving force for populating other parts of OMA, we believe that once the multimedia services become more mature and more widely used in commercial environment, the CORBA community will start looking at possible solutions for the problem of the inclusion of support for these. This is also in line with the view expressed in [111]: ‘as experience in an Application area matures, areas of potential new Common Facilities will be discovered and defined, just as evolving system infrastructures will gradually incorporate pieces of Common Facilities domain into their basic Object Service offerings’.

We believe that there are two possible approaches to the problem of incorporating support for distributed multimedia within the OMA. The first one would be an extension of the CORBA specification to meet the distributed multimedia services requirements. A similar approach has been taken within TINA. For example, the TINA computational model introduces the notion of a stream within its Object Definition Language (ODL) [107]; this effectively being an extension of the CORBA IDL. In addition, within TINA a general purpose binding object has been defined [53] as pointed out in subsection 6.5.8. It is able

36. At time of writing the latest version of CORBA was 2.0.
37. Besides, there are other, more fundamental issues related to distributed computing that need to be addressed first. This includes various interoperability issues, the inclusion of a trading service etc.

to support many communication configurations. The complex system constituting this binding object is referred to as the Connection Management Architecture. The control interface of this binding object is called the Connection Management Interface and is offered by the Communication Session Manager.

The second approach would be to implement basic multimedia services as part of Object Services or Common Facility. For example, within the Common Facility, it would be possible to specify a ‘distributed multimedia service interface’, which would allow users to access a common set of services which embody major multimedia facilities.

6.8 Comments

In this chapter we have illustrated how economic, legal and business concepts can be used to establish a legally valid business contract framework and the corresponding business contract architecture. The following comments will be given to clarify several points in this chapter.

We have discussed how specific characteristics of the parties to a contract, the character of their relationship and the nature of the environment can be used to identify different types of contracts and further what are the most suitable governance structures for these (section 6.2). This discussion was based on a general economic analysis presented in chapter 3 (which can also be a source for a more formal analysis). However, this analysis can be regarded only as a starting framework which provides an insight about the relationship between economic and legal requirements of ODSs and those architectural concepts which are needed to support them. We anticipate that a more rigorous analysis may be needed to address the specific requirements of a particular application or of a system.

Our business contract framework addresses both business level issues and also technology concerns such as QoS. For illustration purposes, we have used fairly simple QoS characteristics in the examples of this chapter. For a more detailed description of QoS parameters, the material of chapter 4 can provide useful guidelines.

In this chapter we have not dealt with the problems of designing efficient contracts, taking into account possible sources of uncertainty. For such an analysis the content of chapter 5 can be used.

In section 6.4 and subsection 6.6.2, we have provided two examples, an electronic stock exchange and the RM-ODP trader. For simplicity of presentation in these examples we have not considered technology driven QoS characteristics but only pure business contract elements. The video service example which illustrates in more detail such QoS details was postponed to subsection 6.7.1, which deals with the problem of positioning these QoS concepts within a generic ODS-A. These two set of examples provide an integrated view on our proposals for the inclusion of the notions of business contract and QoS within an ODS architecture.

Finally, in developing a business contract framework and the corresponding architecture we have tried to map the concepts of contracts from the real world onto an architecture for an ODS. We note that there the nature of the problem domain limits an extent to which such correspondences can be established. As a result, we argue that certain aspects will continue to be handled outside the ODS; examples include subjective judgements such as “intent”, “negligence” and the ultimate authority of society’s legal system.

In addition to the inherent problem of automating the whole spectrum of operations associated with business contracts, another observation is worth making. Namely, the nature of business is such that certain business activities should be left to the domain of human creativity: if everything is automated and standardised, how can competitive edge, often realised owing to the originality of the ideas of entrepreneurs be achieved?

CHAPTER 7

An Implementation of the Business Contract Architecture

7.1 Introduction

The main concepts of the Business Contract Architecture (BCA) developed in the previous chapter provide guidelines for implementing business contract support within an ODS. In this chapter we will describe our implementation which has been developed using a specific ODS architecture: the Orbix platform from IONA Technologies [62], [63], which is a CORBA conformant distributed programming environment. In fact, Orbix is a full implementation of the CORBA 1.0 specification with significant extra functionality added [62].

Our choice of a CORBA conformant platform was motivated by its following features:

- it is an architecture which extends the benefits of object technology across distributed heterogeneous environments: it provides the mechanisms by which objects transparently make and receive requests and responses [109]
- it recognises the importance of support for concepts and services needed to meet changing enterprise requirements of organisations, including support for business objects, common services, type management and the RM-ODP trader functionality

- it is a maturing architecture which is increasingly being accepted commercially, perhaps because it addresses the ODS requirements (as identified in chapters 2 and 3) in a more comprehensive manner than other similar architectures.

In the following sections, the CORBA specification terminology will be used in the context of its Orbix implementation.

7.2 Architectural components

Our implementation of the BCA makes crucial use of both the CORBA Interface Definition Language (IDL) and the Interface Repository (IR), described in subsection 6.6.2.

The IDL specifications of the BCA components' interfaces are included in IDL files which are attached in Appendix A. The code which implements these interfaces is included in Appendix B. We note that the implementation was developed with the purpose of being a proof of concept. It includes all the major concepts and architectural components identified in the previous section and shows that the BCA is indeed feasible, although it is not sufficiently comprehensive to be directly used for commercial purposes.

7.2.1 Contract repository

The content of the Contract Repository (CR) is defined by using the CORBA Interface Definition Language (IDL). The IDL is used to specify both contract elements and contract types (as identified in section 6.4).

1. The contract *elements* can be defined by using the appropriate IDL types supplied. These for example include basic types such as float, double, short, long and any, constructed types such as structures (structs) and template types such as strings and sequences. In our implementation we have chosen to use short, float and string types, which are sufficient to describe contract element types commonly used in a large class of commercial contracts. In addition, any other type can easily be incorporated.

2. The contract *types* can be represented by using the IDL *interface* types. Each interface can contain a variable number of contract elements. Since the CORBA IDL supports inheritance of interfaces, we have chosen this facility as a means to represent relationships between different contract types and thus implement a contract type hierarchy.

An example of an IDL file which contains the description of contract elements, contract types and a simple contract hierarchy is given in the file named *CR.idl*.

The CR can be stored within the CORBA Interface Repository (IR) component, as pointed out in subsection 6.6.2. The use of the IR facilitates dynamic discovery of contract types required by servers or clients, and this has been realised within our implementation. For example, assuming that a server has knowledge about the type of a contract required (e.g. with a name *contractA*) but not its exact structure (which indeed embodies contract semantics), it can access the IR to obtain the structure of a contract template (if such an instance exists within the CR). This involves determining types of contract elements (e.g. a string) and their names (e.g. partyA), by using the IR supplied functions. We have the function, *lookup_name*, which has the name of an object as an input argument (in this case the name of the interface input from the command line) and which returns the IDL sequence *Contained*. This in turn is used to invoke the IR function *describe_interface*, which returns a structure containing the full interface description. This is then used to obtain information about the attributes of the interface (defined in the structure called *AttributeDescription*). These attributes provide information about the types and names of contract elements which in turn describe a *contract template*.

Once a contract template has been obtained, the corresponding values can be supplied. The combination of a contract template and the contract values is called a *contract instance*. In terms of our implementation a contract instance is realised by the IDL sequence type. Each element of a sequence includes two elements: *i)* the IR defined structure which contains a full description of a contract element called *AttributeDescription* and *ii)* an any variable which contains the corresponding contract element value. The latter is needed since the type of a contract element is not known in advance. The contract instance se-

quence is defined in the *CI.idl* file. Within our implementation this contract instance is derived at run time during the server's initiation phase.

Once the server has created a contract instance dynamically, it can then store it internally. Such an instance can be made available to clients on their request and subsequently exchanged between clients and servers and other BCA components. This corresponds to a real life scenario in which fixed contract documents can be made available on users' requests; alternatively, clients can obtain this directly from the CR.

7.2.2 Contract interface

The fundamental contract operations can be implemented in different ways. In order to accomplish a sufficiently generic solution, we have created a *contract interface* which includes support for negotiation, monitoring and enforcing. Such a contract interface can then be used by any server that requires contractual operations: the server's interface can inherit properties of the contract interface, as will be shown in subsection 7.3.1.

The *SRVcontract.idl* file contains the description of the contract interface. It defines the following.

1. A structure (called *Sdata*) containing contract data that are initially offered to each of the clients. This structure includes a contract instance, the name of the contract and two elements which represent a minimum price that server is willing to accept and an upper QoS bound that can be produced. Such a structure is used as a part of the *Client_data* structure that will be instantiated for each of the clients which accepts a server's offer (either the initial offer or after some negotiation). This structure can also include additional data such as name of the client.
2. A set of contract operations to support the following actions.
   a) Negotiation between a server and client (or other third party, such as the Contract Negotiator). These include operations[1] *SRVoffer, CLoffer, SRVresponse* and *Spe-*

1. Due to the lack of a binding type within CORBA, the contract negotiation binding type introduced in subsection 6.5.4 is implemented via these operations.

cialInfo. The first three operations are used in a negotiation between a client and a server and their brief description is given in the *SRVcontract.idl* file. The last operation has a privileged status in that it can be accessed only by designated third-parties (e.g. Contract Negotiator), as will be explained in the next section.

b) Notification of a server about an activity undertaken by a component such as the Contract Monitor or the Contract Enforcer.

7.2.3 Contract negotiator

The role of a contract negotiator (CN) is typically to ensure a more efficient negotiation outcome for one or both parties, based on some information which the parties otherwise would not be able to obtain. In our realisation of a business contract architecture, we assigned the following simplistic role to the CN: find the best deal in terms of the combination <price, QoS> which server can offer to a client. To this end, the CN component's interface (specified in the *CN.idl* file) provides the corresponding IDL operation called *BestDeal*, with an output argument being a contract instance approved by server. Rather than going through several negotiation stages directly with the server, a client can employ services offered by the CN component and thus get the best possible offer which can be obtained from the server. While this simplistic scenario was created only for illustration purposes, we anticipate more complex scenarios reflecting specific business relationships between clients, servers and a CN. These are dependent on particular economic environment and the characteristics of parties as discussed in section 6.2.

7.2.4 Contract validator

The role of the contract validator (CV) is to ensure the use of legally valid contracts. This includes checking of clarity, competence, consideration and legal validity of contract values as described in subsection 6.3.3. Owing to the CR design solution adopted, each of these validity aspects are implicit in the way the CR is implemented. For example, the clarity aspect is implied by the selection of a finite set of contract element types which are used to compose a contract template, the consideration element and the legal validity as-

pects are to a certain extent supported by the fact that contract templates are compiled and stored within the CR by an appropriately authorised authority. The competence element is partly supported by an operation of the CV called *checkReputation* (specified in the file named *CV.idl*). This operation provides a means to check those competence aspects of a server which are related to their previous performance. This operation has the server name as an input argument and a boolean output as a result.

Within our implementation, the CV will check the contract type obtained after negotiation and compare it with the original contract template through the use of the CR. To this end the CV interface provides an operation called *checkValidity*, which has a contract type and a contract instance (previously obtained from the server) as input arguments and an output variable *answer* (which can be true or false, depending on the validity of the submitted contract type and contract instance).

The CV also provides support for storing information about those servers which have not fulfilled contractual obligations in the past. This is achieved via an operation *addServer*, which has a name of server as an input argument and, within our implementation, is invoked by the Contract Enforcer component.

We envisage that the CV component as implemented could be extended with additional validity checking. For example, this could include various security checking procedures, as well as the inclusion of a Legal Rules Repository which can provide additional legal validity checking.

7.2.5 Notary

The role of the Notary is to store contract instances agreed between a client and a server so that they can be accessed at a later stage by either of the parties to the contract or by other third party components. The Notary therefore, plays an important security role in our BCA. We note that the current implementation does not include support for cryptography mechanisms. While this would be essential for the BCA to be commercially acceptable, it is unnecessary for the illustrative purposes of this implementation. There is a wide

choice of commercially available cryptography technologies such as a public/private key technology, which could be included within the implementation in a straightforward manner.

There are two operations associated with the Notary interface: *store* and *retrieve* (these are defined in the *Notary.idl* file). The first operation provides the support to store an agreed contract instance within the Notary. It has a contract instance (of CI type) as an input argument and a new contract instance number as an output argument. The second operation allows retrieval of contract instances based on the contract number previously stored within the Notary. The contract Notary interface and the corresponding data and operations are defined in the file *Notary.idl*.

7.2.6 Contract monitor

The Contract Monitor (CM) has the role of monitoring performance of one or more parties to the contract. This can be requested by both a client (e.g. to monitor the server's QoS delivery) and by a server (e.g. to monitor client's QoS consumption or payments to the server). For the purposes of this demonstration, we have implemented the former operation. This has the name *request* and the following input arguments.

- The number of a contract instance as stored within the Notary.
- The host name of a server.
- The server's name.
- The acceptable QoS degradation of a server (specified by a client).
- Selection of CM's action on detected QoS failure, e.g. whether to record these events for auditing purposes or to activate a Contract Enforcer component (as will be explained in the following section).
- The time when the CM should stop this monitoring activity (its start time is determined by the component which issues the request, e.g. a client).

The CM maintains a list of events which provide relevant information regarding a party's non-performance. This is the IDL sequence type *CMlist*, with elements that correspond to each of the events (named *record).* The CM interface is specified in the *CM.idl* file.

7.2.7 Contract enforcer

The role of the Contract Enforcer is to act upon a detection of a party's failure to fulfil contractual obligations. While this can be applied to both server and client, we have considered only the server's behaviour in our implementation to illustrate the main concepts. To this end the CE component is supplied with an operation named *act_on_server*. It is via this operation that the CM passes information about the host on which server is running and a server's name to the CE. The CE will then send a warning message to a server and subsequently store relevant information about this server in the CV's internal structure so that within subsequent contract validation the server's past behaviour can be checked.

7.3 The use of the BCA by a server and a client

The architectural components realised within our implementation can be used by any server which provides a service and a client which requires a legally sound contractual agreement with the server. In this section we will illustrate how the components realised (and described in the previous section) can be used by servers and clients.

7.3.1 Server side

Assuming that a client has previously found a server which can provide the required information service type, it will typically need to agree on the contract terms with the server. The basic contract operations and structures included within the contract interface can be exploited by the both clients and servers. For example, the contract interface can be inherited through the use of the interface inheritance feature of CORBA IDL: a server providing an information service can simply specify that its interface (called SERVER in this

implementation) inherits contract interface data and operations (as depicted in the file *SERVER.idl)*.

Additionally, a server would need to provide means for the client to invoke the service at the specified time and to define the duration of the service delivery. In terms of our implementation this is achieved through the *service* operations of the server's interface, which has as input arguments the service start and stop time, as well as the name of clients which will be participating in service sessions. The SERVER interface also provides additional operations which allow reading the current server's level of QoS (*getQoS*) and setting it[2] (*setQoS*).

7.3.2 Client side

There may be a number of different scenarios which characterise contract operations and relationships between clients, servers and the BCA components. These depend on the economic, legal and business environment as well as the characteristics of the participating parties. In order to enable the modelling of a different scenarios this implementation provides a means to described the relationships and temporal order associated with the execution of contract operations. This is achieved via the client program module which allows interactive description (via command line) of the desired scenario.

In order to illustrate different types (in economic terms) of clients we have assumed two basic types which we term *easy_going* and *negotiator*. The former will accept an initial server's contract offer, while the latter will be willing to negotiate to achieve better deals. The actions of clients are performed according to the following scenario:

1. Once the client's type is entered (from the command line, when the client is initiated), the client will bind to the interface type which has been previously found[3] and will be

2. This operation is used by a process which simulates QoS variation of a server. The need for a separate process was driven by the lack of a support for threads within the CORBA environment used for the implementation. If threads support were available this QoS variation could have been done internally within the server.
3. At a stage which preceded contractual operations.

presented with the choice to view the server's initial offer (via *SRVcontract* interface *SRVoffer*).

2. After viewing the initial offer and if the client's type is negotiator, it can then be engaged in further contract negotiation with the server. In addition the client will have a choice of doing the negotiations directly with the server, or using the services of a third-party Contract Negotiator, which can provide a more efficient negotiation outcome (from the client's point of view) but at a price. In our implementation the negotiation process is centred around contract values, although a more general negotiation can be realised, which would include negotiation related to the addition or deletion of certain contract elements.
3. Once the negotiation is complete, the client can validate the contract instance which has been agreed upon. This is facilitated using the CV component. In the context of our implementation the contract validation is based on using the *checkValidity* and *checkReputation* operations of the CV.
4. Having successfully completed validation, this contract instance can then be stored within the Notary component, via the supplied operations.
5. After the above contract establishment steps are undertaken, the client can issue a request to the server to start the service at specified time, and to stop the service delivery after the specified time interval. These times are entered interactively from the command line during the client's execution.
6. Upon the server's start of service execution, the client can issue a request to the CM to start monitoring server's performance. Within our implementation this is reduced to the monitoring of QoS that's being produced by the server. The subsequent monitoring and enforcing activities are assigned to the CM and CE components, as explained in the previous section.

7.4 Some examples of different contract life time scenarios

In this section we will outline several examples of the use of our BCA implementation which correspond to some typical contracting scenarios. These are accompanied by the corresponding output files as appended in Appendix C. We note that there may be many

other scenarios with different relationships between the parties and different temporal orderings.

This description is based on the contract life time stages identified in section 6. 3.

7.4.1 Contract establishment stage

In each of the following examples it is assumed that the server components (e.g. CV, Notary, SERVER etc.) will be started first. The client program can then be used to select a particular choice of components and actions required.

Example 1: no negotiation, use of the CV and Notary components

The file *SERVERout1* outlines the steps through which a server passes during its initialisation. As can be seen from this file, the first step is binding to the IR, after which the server can dynamically create a contract template. In this example the server has found that the contract name is *contract* and it has 6 contract elements: *partyA* and *partyB* of type string, consideration elements being the float variables *partyA_gives* and *partyB_gives* and two string variables with the names *date_of_agreement* and *domain_name*. We envisage that a contract template such as this can be sufficiently generic to be applicable to any type of business contracts (recall the *CR.idl* file which stores the contracts type hierarchy). Additionally, during the initialisation stage the values of the contract are interactively entered (these are depicted in the *SERVERInput* file[4]). The contract instance derived is also printed. The last step in the server's initialisation is the definition of the factors which represent QoS and price tolerance by the server.

The file *CLIENTout1* depicts the steps which a client program undertakes. The first step is binding to the server's interface, upon which the client can check the initial offer. The client can then have the choice of validating the contract instance obtained from the server against the type of the contract (also obtained via the server). This involves the use of the

4. These are arbitrary numbers. In a real life scenario they will represent the appropriate values such as price in monetary units and measure of QoS (as discussed at length in chapter 4).

CV component: initially binding to it and then invoking the CV operations *checkValidity* and *checkReputation* as described in subsection 7.2.4. If successful, this contract instance can then be stored within the Notary. The files *CVout1* and *Notaryout1* illustrate some actions associated with the operations of the CV and the Notary.

Example 2: direct negotiation, use of the CV and Notary components

This example illustrates direct negotiation between a client and a server. To this end the client (which is of the type *negotiator*) chooses to be engaged in direct negotiation with the server. After viewing the initial server's offer, the client first offers a <QoS, price> pair, i.e. <12,70>, as depicted in the file *CLIENTout2*. Since these are not acceptable by the server (as also depicted in the file *SERVERout2*), the client will come up with another counter-offer. in this example the deal is negotiated after the third negotiation stage.

Example 3: negotiation with the CN component, use of the CV and Notary components

In this example the client chooses to use the CN component. Owing to the fact that a CN has access to the information about the server's tolerance factors regarding QoS and price (which the client did not know in the previous example), there will be only one negotiation step, and the CN will be able to find the best deal (which is equal to the maximum QoS that the server can provide and minimum price that it is willing to accept for it). This is shown in the file *CLIENTout3*.

The file *CNout3* depicts the major actions of the CN. As can be seen from this file, the CN determines that the tolerance factors for QoS and price are 1.2 and 0.8 respectively and provides the counter offer to the server with these values. Although this counter offer cannot be regarded as the best deal from server's point of view, it is still acceptable and the server would accept it.

Note that the contract type supported by the server is *contractB* in this case. It is a contract

which is inherited from the base contract type, *contract*, and adds two additional string types, *partyAagent_name* and *partyBagent_name*, which represent agents for *partyA* and *partyB* respectively (this is shown in the *SERVERout3* and the *CVout3* files).

7.4.2 Contract performance stage

These examples illustrate contract operations after the contract establishment steps have been successfully performed.

Example 4: contract monitoring through the CM component

This example demonstrates how the client can (after checking validity and storing the contract instance in the Notary) invoke the required service as previously negotiated. The client typically specifies the start time of the service and its length. This is interactively entered from the CLIENT's command line (e.g. 1 min. and 5 mins as depicted in the file *CLIENTout4*). Upon the server's start, the client can also choose to invoke operations of the CM. In this example the client chooses to do no more than to record non-performance of the server (in the next example it also requests some enforcing actions).

To illustrate the concepts of our BCA, we have chosen to simulate QoS variation of the server. In this example the variation was chosen to be a step function[5], in which 90% of the time the server produces agreed upon QoS (with the value of 10) and 10% of the time its QoS is below the acceptable threshold (which is chosen to be 9, as specified in the file *SimInput*) and has the value of 8. This has been achieved via the auxiliary component called *Sim*, whose sole purpose is in changing the server's QoS variable. The actions of this simulator are depicted in the file *SIMout4*.

At the times when the QoS degradation is below the threshold, the CM (which previously obtained information from the Notary about the contract instance agreed between the server and the client) records the degraded QoS value, the time when this happened and

5. Any other function which model some QoS delivery can be used.

other relevant details. These are then stored within the CM internal structure CMList, pointed to by the pCMLIST pointer. The actions of the CM are depicted in the file *CMout4*.

The operations in this example also include activities of the CV and Notary (as shown in Files *CVout4* and *Notaryout4* respectively).

Example 5: contract monitoring through the CM and the use of the CE component

This example can be regarded as an extension of the previous example. It involves the use of the CE component, which is invoked by the CM upon a detection of a QoS degradation (as specified during client's initiation phase, file *CLIENTout5)*. The additional activity which the CM performs is invoking the operations of the CE (as shown in the file *CMout5*). The CE then sends a warning message to the server (as shown in the file *SERVERout5*) and stores the information about the server's non-performance (e.g. its name) within the CV's internal structure, *CVlist* (as can be seen from the file *CEout5* and also in the corresponding CV's action recording, the file *CVout5*).

7.4.3 Post-contract phase

The following example shows how information about non-performance of a party to the contract can be used in subsequent business dealings by any future potential trading partner.

Example 6: contract monitoring through the CM and the use of the CE component

The records about the server's non-performance are stored in the CV's internal structure. This information can then be used by clients when checking the server's reputation in future. The files *CLIENTout6* and *CVout6* illustrate how a client can obtain information

about the server's non-performance (recorded during the previous contract performance stage, such as the one in example 5) and how this can be used to terminate further contract negotiations with a server.

7.5 Comments

We note that the example scenarios presented previously are only a representative sample of operations in the contractual relationships between clients and servers which occur in a real world business environment. There are numerous different relationships between commercial partners as well as various temporal orderings of contactual operations, both of which can be significantly more complex than those presented here. The examples chosen however, prove the usability of the BCA and they are sufficient to show that the BCA is feasible and can be developed using a commercially available distributed environment.

Our choice of a CORBA conformant distributed programming environment to be used as a prototype for testing the main architectural concepts of the thesis was driven by the CORBA features mentioned in section 7.1. We exploited these features to provide support for the business related concepts of our BCA. However, as pointed out in section 6.7, current CORBA implementations do not have mechanisms to measure and control some technology-related QoS characteristics, such as those required for distributed multimedia services[6]. We anticipate that future versions of CORBA (or services within the OMA Reference Model) will incorporate such mechanisms, in response to the commercial requirements for multimedia services. Once this is available, it will be possible to extend this implementation towards support for the distributed multimedia services.

6. Neither did any other environment available at the time of undertaking this research.

CHAPTER 8

Conclusion

In this chapter the conclusions of the thesis are outlined. The chapter first presents the main argument of the thesis and the problems identified. Following this, major contributions are summarised, including both broad findings and more specific results. We also propose several future research directions which can further broaden or deepen the work we have undertaken.

8.1 Problems identified

In this thesis we have investigated the use of specific economic theories and the relevant concepts from business and law to address and solve several enterprise related concerns of open distributed systems (ODSs).

We have chosen to restrict the problem domain of the thesis from a broad set of ODS enterprise concerns (which are outlined in chapter 2) to specific economic and business aspects. The motivation for such a focus was the fact that these issues are of particular significance for the commercial success of ODSs in terms of their more rapid and wide deployment, and

their evolution towards a ubiquitous, global information infrastructure.

Our starting premise was the fact that the enterprise related issues are emerging as an important topic for end-users, IT managers, strategic planners and designers of ODSs, and yet many of these issues are not appropriately understood. Clearly, the importance of economic and business aspects of ODSs over last several years is a result of the increasingly competitive character of the market for IT services and products. Those who invest in the latest technology such as ODSs expect to gain from its new capabilities in terms of advancing their enterprise objectives. At the same time, they are also more selective when choosing a manufacturer or a service provider, thus urging suppliers to improve the quality of their offerings. For suppliers, this typically means gaining a better understanding of the environment in which ODSs are to be implemented, a greater appreciation of the purpose that they will serve for end users, and incorporating these facets into the implementations of services or systems.

We began this research with the identification of broad ares of enterprise features inherent in ODSs and services in ODSs: those that are new in comparison to traditional distributed systems. We recognized the following four specific classes of problems that we felt needed to be further studied.

First, the new economic and business related features of ODSs, such as *i)* a possibility of an open access to the services and resources within a system (which can lead to an interaction between a large number of economic entities with different objectives and requirements), and *ii)* the capability to facilitate provision of services across geographical, administrative and other boundaries, suggest that ODSs resemble large economic systems which are, from economic point of view, *quite different* and *more complex* than traditional distributed systems. This deserves an identification of suitable paradigms from economics and business which can be used to *explain, predict* and *quantitatively model* interactions between the participating economic entities. In addition, the new technological capabilities of ODSs set an agenda for investigating their impact on economic institutions, such as markets and organisations.

Second, we found that there was no consensus (there are often even disagreements) about the *meaning* of Quality of Service (QoS) in ODSs, and how it should be *measured*. This problem is of major concern to the ODS community[1], especially when taking into account the facts that QoS represents an important economic parameter and that it is relevant to a broad range of different services. To address this problem, a generic QoS framework is needed, which would facilitate the description and measuring of QoS in a user-oriented and service-independent manner. This would help to reach a consensus about QoS treatment, so that users can cope with a large number of existing service types and the proliferation of new services.

Third, due to a number of new technological, organisational and human factors which arise in ODSs, the problem of *uncertainty* can arise, with potential ramifications on service (or QoS) delivery in such systems. This problem is implicitly recognised in a number of individual areas, such as the problem of accurately expressing the semantics of services based on the users' view, the problem of searching for services in the global information marketplace and various security issues. However, not much work has yet been done to understand the *economic* effects of uncertainty in an open information market, for example, potential economic losses that some parties can experience (say) due to the opportunistic behaviour of others.

Fourth, despite one of the major goals of ODSs being to provide interworking across organisational boundaries, we found no architectural model which can be used to extend current ODS architectures to support *electronic contractual business dealings*. While such an architecture may be premature when considering today's state of commercial ODS systems, we anticipate that, from the point of the research community, initial work in this direction would be beneficial, especially in terms of promoting interactions with the legal community. This would provide two important benefits. First, by understanding the economic and legal bases of contracts, a computer scientist can propose an initial architecture which will support electronic business contract dealings. Second, the experts from commerce and law can then provide their input and feedback to such an architecture so that a sound, legally valid business contract architecture can ultimately be developed.

1. As raised in discussions at the QoS workshop held as a part of the ICODP'95 conference [94].

This will hopefully foster electronic business contract operations between enterprises and a more active involvement of the legal profession (which is currently lagging behind new technologies). Such a synergy would facilitate solving a number of sensitive legal issues that present a serious obstacle to widely implementing electronic commerce interactions.

8.2 Contributions

The nature of these four problem areas has clearly indicated the need for seeking solutions in economics, as well as in concepts from business and law. Two categories of the contributions of the thesis can be distinguished. First, as part of the exploratory aspect of the thesis, we have identified a number of relevant economic theories and notions from business and law. We have then studied how they can be applied in each of the four problem areas. This category is summarised in subsection 8.2.1. Second, we have obtained more specific results by applying some of these ideas in particular areas. This includes both the theoretical and practical contributions, as summarised in subsection 8.2.2.

8.2.1 Broad findings

1. We have recognised that the enterprise issues in ODSs require consideration of the concepts from sciences such as *economics*, *sociology*, *psychology* and *organisational theory*. This has been supported by a number of examples which illustrated how non-technological factors related to these disciplines should be accounted for when designing an ODS that would best fit the purpose for which it is intended (chapter 2). We have also shown how various enterprise issues can be related to a framework of the RM-ODP, in particular the RM-ODP enterprise viewpoint.

2. We have found, and with a number of examples illustrated, that it is worthwhile applying *appropriate economic theories* to the corresponding enterprise ODS issues (chapter 3). Following this economic spirit we have found both positive and normative aspects to be pertinent. The positive aspects can be used for example to *explain* current changes influenced by computing and telecommunication industries within

organisations and markets, including the growing importance of QoS as an economic variable and also to *predict* the responses within these institutions to emerging ODS technologies. For example, this can be used for the structuring of an underlying information system to best fit the structure of an evolving organisation. The normative aspects can be used to arrive at a quantitative model of relevant behaviour of interactions in an ODSs. In particular, suitable modelling tools used in economics, such as game theory, can be applied in that context. We have shown one such modelling application, namely the principal-agent model which can be used to model interactions between a user and a service provider in an uncertain environment.

8.2.2 More specific results

1. We have applied particular economic theories to solve several *specific* enterprise related problems in ODSs. In particular, we have used the following economic theories.

   a) *Lancaster's theory of consumer demand* has been used as a point of departure in developing a general framework for thinking about and describing QoS in ODSs. The importance of this framework (that has been developed in chapter 4) is its capacity to alleviate current problems in the ODS community related to the lack of consensus on the meaning of the notion of QoS. While we accept that QoS specification and measures are dependent on the types of distributed services, we argue that it is critical to adopt a consistent, and service-independent way of describing and measuring QoS. This should be based on the users' view of the major quality aspects of a service that directly influence their enterprise objectives (which can formally be expressed by means of appropriate utility functions). We have proposed Lancaster's theory as suitable for developing such a framework.

   b) *Game theory,* in particular the principal-agent model has been used to solve the problem of uncertainty which can arise in ODSs due to the unpredictability of the environment and asymmetric information (chapter 5). We have first located different sources of uncertainty in an ODS. We have then shown the ramifications of

uncertainty on efficient and equitable distribution of benefits from the interactions among parties in ODSs. In particular, uncertainty can impact service delivery so as to reduce the economic efficiency of certain parties (e.g. end-users who cannot fully observe actions of service providers). We have shown how the principal-agent model can be used to quantitatively model interactions between a user and a service provider in an ODS. The model can then be used to design efficient contracts, which provide *i)* incentives to the agent to perform optimally from the user's point of view and *ii)* sharing of the risk arising from environmental uncertainty.

c) *Transaction cost economics* has been used to identify the types of services which can be particularly affected by uncertainty (chapter 5). This has been done by treating service delivery in an ODS as a special kind of economic transaction and then applying the main results of transaction cost economics. This theory has also been used to provide an insight into different types of contracts that can be used in ODSs (chapter 6) as well as in providing guidelines on how to structure an ODS system according to the institutional structure (organisations or markets) in which it will be utilised (chapter 3). Additionally, transaction cost economics has been used to analyse how ODSs can reduce the transaction costs of enterprises and how this can impact their structure.

2. We have also recognised the potential semantic difficulties in basing QoS description on user perceptions. These difficulties are particularly acute because information services are harder to conceptualise than other, everyday services, especially when taking into account the lack of users' previous experience. Being aware of numerous human issues associated with accurately defining and measuring QoS, we have proposed the use of techniques from *market research* and *consumer behaviour* as a part of a QoS framework. We have suggested two such methodologies which can be used in this context; conjoint analysis and information integration theory.

3. Relevant concepts from *business* and *law* have been incorporated into the ODS enterprise domain. We recognised that the use of the aforementioned economic theories addresses one aspect of enterprise related problems and that economic theories can be

exploited once the ODS technologies are mature enough for implementation. However, we also recognised that for commercial success of an ODS architecture, one needs to provide concrete *architectural mechanisms* which will facilitate and promote ODSs in terms of their two important goals:

a) to make the benefits of distributed systems accessible to a wider user community

b) to provide an effective sharing and an efficient utilisation of information services and computational and communication resources across organisational boundaries.

We consider support for *business contracts* to be one of the critical mechanisms required to foster the above goals. To this end we have endeavoured to provide a support for business contracts within ODS architectures. In addition to understanding contracts from an economic standpoint, one also needs to understand them from legal and business perspectives. This is needed as a prerequisite in establishing a sound business contract framework and the corresponding *business contract architecture* that can be used to extend the capabilities of current ODS architectures. Such an architecture has been developed and its concepts have been positioned in relation to a generic ODS architecture (chapter 6). We have successfully implemented the core of this architecture (chapter 7), and this prototype model has demonstrated the feasibility of the business contract architecture proposed.

In the course of testing various hypotheses throughout the thesis, we have consistently used an example of the RM-ODP trader service. Motivation for this was in the fact that the trading service will represent an important component of an ODS and that it is rich enough to serve as a good example for illustration purposes. In addition, we have used several examples of the application domain. These are a multimedia-oriented, tourist information service, as an example for illustrating the use of the QoS framework suggested, an electronic stock exchange to demonstrate the positioning of the business related concepts within our business contract architecture, and video service to explain architectural support for QoS pertinent to distributed multimedia services.

8.3 Future work

We believe that this thesis has highlighted a number of interesting and (for the commercial world) important concerns, and can be seen as an initial step in providing guidelines and solutions for the class of ODS enterprise problems of an economic and business nature identified in the course of this research. It can also serve as a good starting point to further investigate a number of specific concerns, as follows.

We anticipate that a significant research still remains to be done in the area of *QoS specification* and *measuring* with the aim of better understanding and incorporating the user view on different QoS issues. This should be based on an inter-disciplinary approach including the use of recognised techniques from the fields of consumer research, psychology and quality management. In relation to this, it would be worthwhile to apply the QoS framework and related QoS metrics developed in chapter 4 to a specific service. This was not feasible within the time frame of this research, but we plan to do so in the context of a local telecommunication service.

In relation to agency theory modelling, additional investigations are required for finding suitable mechanisms for ODSs which can facilitate *on-line* implementation of the principal-agent modelling approach. This includes the study of mechanisms by which parties to interactions can reveal their characteristics to the system so that a numeric on-line algorithm can be used to dynamically adjust to the possible changing behaviour of parties. In addition, more extensive research is required to model the influence of many agents or many principals on the QoS delivery in ODSs and we believe that future results from agency theory will be most beneficial in this context.

The principal-agent model represents one particular class of problems which are within the broader scope of *game theory*. We anticipate that game theory can be used in a wider context, to address a number of problems related to the interactions between parties in ODSs. In particular, those problems whose essence is in finding such rules which would accomplish certain goals while assuming either the non-cooperative or the cooperative behaviour of players. While the principal-agent problem belongs to noncooperative game theory, there can be other issues where cooperative game theory can be applicable. For

example, solving the problems of interworking, such as federation between the trader services belonging to different administrative domains.

In general, we found that the complexity of the problems associated with ODSs is such that there is a limit to which current economic disciplines and theories can be applied to model and analyse these systems. The number of variables that can be involved in such a system, their dynamic nature and the lack of full information about those variables can lead to situations in which it can be hard (if not impossible) to obtain exact solutions. It is our belief that future contributions from economics can be most valuable to further address many of the economic related problems in ODSs which have been identified in this thesis. In addition, we believe that the capabilities of ODSs (e.g. better monitoring and improved accuracy of information) can provide a basis for the eventual testing of the economic theoretical models, such as numerous variants of functionality of markets. This, of course, is conditioned by the level of prices associated with ODS technologies

As part of future work, it is envisaged that further implementations of the concepts of the *business contract architecture* (as provided in chapter 7), can be used as a commercial test of the usability of this framework and identification of potential problems associated with it. Perhaps, the most important short term effort is the incorporation of the relevant *security* issues as they gradually mature. Additionally, the commercial availability of *distributed multimedia services* and mechanisms to control and measure the related QoS characteristics can provide a foundation for further testing of the capabilities of our business contract architecture to support the incorporation of the technology oriented QoS details into business related issues. We also anticipate that contributions from the field of artificial intelligence can be applied so that certain *human* decision making can be automated.

Finally, new characteristics of global information service markets are opening new questions for economics, as well as related sciences. For example, relevant work from this thesis can lead to further research to provide new insight into how ODSs can change the structure of organisations and markets. This will require a multi-disciplinary approach, including disciplines such as management science and organisational theory.

Appendix A: IDL files

A.1 CR.idl

```
// - should be compiled with the IR preprocessor -R switch. This
//   stores this file within the IR
// - contains different contract templates, each of which can have
//   different numbers of contract elements
//
// - includes only attributes
//            operations are implicit:
//
//            'store' ->    by creation of interface definitions
//                          and their relationships and storing them
//                          into the Interface Repository (IR)
//
//            'retrieve'->  by using IR operations supplied
//
//
//            Note: this simulates a very primitive type management system
//
//            comment: deals with two-party contracts
//

interface contract {
              readonly attribute string partyA;     // e.g. Service Provider
              readonly attribute string partyB;     // e.g. User
              readonly attribute floatpartyA_gives; // e.g  SP's QoS value
              readonly attribute floatpartyB_gives; // e.g. User's payment
              readonly attribute string date_of_agreement;
              readonly attribute string domain_name;
              };

//  some inheritance (for illustration purposes)

interface contractA : contract{
              readonly attribute stringsettlement_date;
              };

interface contractB : contract{
              readonly attribute stringpartyA_agentname;
              readonly attribute stringpartyB_agentname;
              };
```

A.2 CI.idl

```
#ifndef _CI_idl_h
#define _CI_idl_h

#include <AttrDf.idl>    // IR's AttributeDescription  defined here

// An instance of a contract element. This includes:
//    - contract element parameters: type(long,float,string), name
//    - value to be associated with CE

struct CEInstance {
              AttributeDescription CE;
              any CEvalue;
}; // CEInstance

typedef sequence<CEInstance, 20> CI; // contract values

#endif
```

A.3 SRVcontract.idl

```
#ifndef _SRVcontract_idl_h
#define _SRVcontract_idl_h

// This idl file specifies contract operations of a server

#include "CI.idl"    // Contract Instance sequence defined here

// SERVER's data (same data to be offered initially to each client)
struct Sdata {
              CI m_CI;
              string m_contract_type; // mult.factor for max QoS
              float m_QoS_limit;      // mult.factor for min price
              float m_price_limit;
};
typedef Sdata SD;

// Client's data structure (one instance per client)
struct Client_data {
SD contract_data;
short m_Saccept;
string Cname;
};
typedef sequence<Client_data> CL_data;

interface SRVcontract {
//            ******** OPERATIONS **********

//      -------- NEGOTIATION ---------

// The following operation enables CLIENT to view SERVER's offer.
// No input parameters (assumption 1a).

void SRVoffer (out CI x, out string contract_type);

// through the following operation CLIENT can inform SERVER about
// its agreement (Caccept = T) or counteroffer

void CLoffer (in short Caccept, in CI counteroffer);

// to check SERVER's response to the Cleint's counteroffer

void SRVresponse (out short Saccept);

// for access to a special SERVER's information, which third parties
// such as Contract Negotiator or Contract Arbitrator can access

void SpecialInfo(in string name, out float QoS_factor, out float
price_factor);

//      -------- MONITORING ---------

// for notifying SERVER about monitoring actions (e.g.start of monitoring)

oneway void SRVmonitor (in string Mmessage, in string start_time);
```

```
//      -------- ENFORCING ---------

// for warning SERVER about possible actions (e.g. that enforcing can result)

oneway void SRVenforce (in string Emessage);
};
#endif
```

A.4 CN.idl

```
// This idl file specifies Contract Negotiator's interface
//

#include "CI.idl"    // Contract Instance sequence defined here

interface CN {

//             ******** OPERATIONS **********

// find the best possible deal which can be obtained from a server

void BestDeal (out CI Best);

};
```

A.5 CV.idl

```
// This idl file specifies Contract Validator's interface
//
//

//

#include "CI.idl"    //  IR's AttributeDescription  defined here

typedef sequence<string> BadServers;

interface CV{

//             ******** OPERATIONS **********

void checkValidity (in CI x, in string contract_type, out short answer);

void checkReputation (in string Server_name, out short Rep_answer);

void addServer (in string Server_name);


};
```

A.6 Notary.idl

```
//
// Provides two operations:
//
//            - STORE a contract instance agreed by the CLIENT and SERVER
//                           - This adds a contract number to a contract
structure
//                           - Stores instances in a list
//
//            - RETRIVE a contarct instance, based on the contract number
//
//            Note:         the use of dynamically cretaed sequence CI
//                          ensures that any type of contract template can
//                          be passed by a Notary's client (CLIENT,
//                          SERVER or others)

#include "CI.idl"    //  IR's AttributeDescription  defined here

typedef sequence <CI> CIList; // contract values

interface Notary {

//            ******** ATTRIBUTES **********

              readonly attribute CIList NL ;

//            ******** OPERATIONS **********

// store a contract instance and create its number which will be passed to
// contract monitor (CM)

void store (in CI x, out unsigned long No);


// retrieve the contract instance

void retrieve (in unsigned long ContractNo, out CI Instance);

};
```

A.7 CM.idl

```
struct record{
              unsigned long CoNo;          // what is Contract Number
              string host;                 // at which host is ...
              string server_name;          // ... this server
              float  QoSvalue;             // what is real QoS value
//            string time;                 // when this happened
}; // record

typedef sequence<record> CMlist;  // list of recorded values


interface CM {
//            ******** ATTRIBUTES **********

              readonly attribute CMlist CML ;

//            ******** OPERATIONS **********

oneway void request (in unsigned long CoNo, in string host, in string nameS,
              in float QoSdegrad, in string what_to_do, in unsigned long stop
);

};
```

A.8 CE.idl

```
struct CErecord{
               unsigned long CoNo;          // what is Contract Number
//             string host;                 // at which host is ...
//             string server_name;          // ... this server
//             float  QoSvalue;             // what is real QoS value
//             string time;                 // when this happened
}; // record

typedef sequence<CErecord> CElist;  // list of recorded values


interface CE {

//             ******** ATTRIBUTES **********

               readonly attribute CElist CEL;


//             ******** OPERATIONS **********

void act_on_server (in string message, in string host, in string nameS );

};
```

A.9 SERVER.idl

```
#ifndef _SERVER_idl_h
#define _SERVER_idl_h

//
// SERVER inherits attributes and operations of contract interface
//

#include "SRVcontract.idl"

typedef sequence<float> QoSvalues;

interface SERVER : SRVcontract {
  void getQoS (in unsigned long ClNo, out float currentQoS);
  oneway void setQoS (in unsigned long ClNo, in  float currentQoS);
  oneway void service (in unsigned long start_time, in unsigned long
stop_time, in string CLname);
};

#endif
```

A.10 Sim.idl

```
#ifndef _Sim_idl_h
#define _Sim_idl_h

//
typedef float QoS;
typedef sequence<QoS, 10> QoS_sim;

interface Sim {

oneway void step (in unsigned long ClientNo, in unsigned long stop, in short
change, in short drop);

};

#endif
```

Appendix B: Implementation files

B.1 SRVcontract_i.h

```
#ifndef __SRVcontract_ih
#define __SRVcontract_ih

#include "SRVcontract.hh"
#include "utils.h"


class SRVcontract_i : public virtual SRVcontractBOAImpl {
protected:
        SD m_SRVinit;
              CL_data* pCL_data;

public:
              // constructor
        SRVcontract_i (const CI& x,  char* name, float QoSlimit,
float pricelimit);

              virtual void SRVoffer (CI& x, char *& contract_type,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void CLoffer (short Caccept, const CI& counteroffer,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void SRVresponse (short& Saccept,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void SpecialInfo (const char * name, float& QoS_factor,
float& price_factor, CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void SRVmonitor (const char* Mmessage, const char*
start_time,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void SRVenforce (const char * Emessage,
CORBA::Environment &IT_env=CORBA::default_environment);

};

#endif
```

B.2 SRVcontract_i.C

```
#include "SRVcontract_i.h"
#include <stream.h>
#include <stdlib.h>
#include <unistd.h>

SRVcontract_i::SRVcontract_i (const CI& x,  char* name, float QoSlimit,
float  pricelimit) {

  // initialising SERVER's first offer (to be offered to every client)
m_SRVinit.m_CI = x;
m_SRVinit.m_contract_type = new char[50];
strcpy (m_SRVinit.m_contract_type, name);

m_SRVinit.m_QoS_limit = QoSlimit;
m_SRVinit.m_price_limit = pricelimit;

// Create a pointer to a sequence with Clients' contract data
pCL_data = new CL_data (20);
pCL_data->_length = 0;   // always points to the last used !!
pCL_data->_maximum = 20;
} // end constructor

void SRVcontract_i:: SRVoffer (CI& x, char *& contract_type,
CORBA::Environment &IT_env) {

  // get SERVER's initial offer
contract_type = new char[50];
strcpy(contract_type, m_SRVinit.m_contract_type);
x = m_SRVinit.m_CI;
}


void SRVcontract_i :: CLoffer (short Caccept, const CI& counteroffer,
CORBA::Environment &IT_env) {

  // inform SERVER about contract's acceptance or a counteroffer
float ClQoS, Clprice;

if ( Caccept == 1 )
  {
    cout << "CLIENT accepted: SERVER to start executing service" << endl;
    cout << "QoS variable may need to be checked during service execution ! "
      << endl;

// Create a CL_data sequence for new client:

    (pCL_data->_length)++;
    pCL_data->_buffer[pCL_data->_length].contract_data = m_SRVinit;

    if (pCL_data->_length >pCL_data->_maximum ){
              cout << "Initial number exceeded !! " << endl;
              pCL_data->_maximum = pCL_data->_length;
              }
  }
```

```
else
  {
    pCL_data->_buffer[pCL_data->_length].m_Saccept = 0;
    cout << "SERVER reads counter-offer and finds that it is: " << endl;
    printCI(counteroffer);
    cout << endl;
   cout << "SERVER makes decesion whether to accept counter-offer or not .. "
      << endl << endl;
    sleep (2);

    float min_price, max_QoS;

    min_price = *( (float*) m_SRVinit.m_CI[3].CEvalue._value )
      * m_SRVinit.m_price_limit;
    max_QoS =  *( (float*) m_SRVinit.m_CI[2].CEvalue._value )
      * m_SRVinit.m_QoS_limit;
    cout << "Since SERVER accepts min. price of " << min_price
      << " and can provide max. QoS level of " << max_QoS << endl;

    ClQoS   = *( (float*) counteroffer[2].CEvalue._value );
    Clprice = *( (float*) counteroffer[3].CEvalue._value );
    cout << "... and since Client's QoS offer is  " << ClQoS
      << " and price offer is " << Clprice << endl;

    if ( ClQoS <= max_QoS && Clprice >= min_price )
      {
     cout << "SERVER will accept  and will start executing service " << endl;

      (pCL_data->_length)++;
      pCL_data->_buffer[pCL_data->_length].contract_data = m_SRVinit;
      pCL_data->_buffer[pCL_data->_length].m_Saccept = 1;

    if (pCL_data->_length > pCL_data->_maximum ){
      cout << "Initial number exceeded !! " << endl;
      pCL_data->_maximum = pCL_data->_length;
    }

      }
    else{
      cout << "SERVER not accepted-> up to Client to proceed ... " << endl;
    } // else

  } // else

} // CLoffer



void SRVcontract_i:: SRVresponse (short& Saccept, CORBA::Environment &IT_env)
{
Saccept =  pCL_data->_buffer[pCL_data->_length].m_Saccept;
} // SRVresponse



void SRVcontract_i:: SpecialInfo (const char * name, float& QoS_factor,
float& price_factor, CORBA::Environment &IT_env) {
```

```
// ensure that only CN has access to this privilidged information
if (strcmp (name, "CN") == 0 ){
  QoS_factor =  m_SRVinit.m_QoS_limit;
  price_factor = m_SRVinit.m_price_limit;
   }
} //SpecialInfo


void SRVcontract_i:: SRVmonitor (const char * Mmessage,
const char * start_time, CORBA::Environment &IT_env) {
cout << "The following message arrived from Contract Monitor " << endl;
cout << Mmessage << endl;
cout << start_time << endl;
}

void SRVcontract_i:: SRVenforce (const char * Emessage,
CORBA::Environment &IT_env) {
cout << "The following message arrived from Contract Enforcer " << endl;
cout << Emessage << endl;
}
```

B.3 SERVER_i.h

```
#ifndef __SERVER_ih
#define __SERVER_ih

#include "SERVER.hh"
#include "SRVcontract_i.h"

#include "utils.h"

class SERVER_i : public virtual SERVERBOAImpl, public virtual SRVcontract_i {
protected:
        float m_QoSInit;
              QoSvalues* pQoSlist;// current QoS values lsit (randomly
change)

public:
              // constructor
    SERVER_i (const CI& x, float QoSInit, char* name, float QoSlimit, float
pricelimit);

              virtual void getQoS (unsigned long ClNo, float& currentQoS,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void setQoS (unsigned long ClNo, float currentQoS,
CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void service (unsigned long start_time, unsigned long
stop_time, const char * CLname, CORBA::Environment
&IT_env=CORBA::default_environment);

};

#endif
```

B.4 SERVER_i.C

```
#include "SERVER_i.h"
#include "Sim_i.h"

#include <time.h>

#include <stream.h>
#include <stdlib.h>

#include <unistd.h>


SERVER_i::SERVER_i (const CI& x, float QoSInit, char* name, float QoSlimit,
float pricelimit) : SRVcontract_i (x, name, QoSlimit, pricelimit) {

// Create a pointer to a sequence with Clients' QoS data

pQoSlist = new QoSvalues (20);
pQoSlist->_length = 0;   // always points to the last used !!
pQoSlist->_buffer[0] = QoSInit;   // always points to the last used !!
m_QoSInit = QoSInit;
pQoSlist->_maximum = 20;

} // end constructor


void SERVER_i :: getQoS (unsigned long ClNo, float& currentQoS,
CORBA::Environment &IT_env) {
currentQoS = pQoSlist->_buffer[ClNo];
}

void SERVER_i :: setQoS (unsigned long ClNo, float currentQoS,
CORBA::Environment &IT_env) {
pQoSlist->_buffer[ClNo] = currentQoS;
}



void SERVER_i:: service (unsigned long start_time, unsigned long stop_time,
const char * CLname, CORBA::Environment &IT_env) {

time_t tiempo;

Sim* pSim;
struct tm* T;
unsigned long startT;
unsigned long stopT;

float factor, firstQoS;
short time_of_change, drop;

// Create a QoS value for new client:

    (pQoSlist->_length)++;
cout << "pQoSlist->_length = " <<  pQoSlist->_length << endl;
```

```
tiempo = time(NULL);
cout << " It is now " << ctime (&tiempo) << endl;

tiempo = time(NULL);
T = localtime(&tiempo);

if (T->tm_min >= (60-start_time))
  startT = T->tm_min -(60-start_time);
else
  startT = T->tm_min + start_time;

cout << "startT = " << startT << endl;

if (startT > (60-stop_time) )
  stopT = startT -(60 - stop_time);
else
  stopT = startT + stop_time;

cout << "stopT = " <<  stopT << endl;
cout << endl;

tiempo = time(NULL);
T = localtime(&tiempo);

while (! (T->tm_min == startT) ) {
  cout << " T->tm_min = " << T->tm_min << " T->tm_sec = " << T->tm_sec << "
startT = " << startT << endl;
  sleep(10);
  tiempo = time(NULL);
  T = localtime(&tiempo);
}
  cout << "Start producing QoS now ... " << endl;

/// new stuff from here
 cout << " try to bind to Sim .... " << endl;

  TRY {
    pSim = Sim ::_bind(":SIM", "foxtail.dstc.edu.au", IT_X);
    cout << "SERVER bound to sim "  << endl;
  } CATCHANY {
    cerr << "Bind to SIM  object failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(1);
  } ENDTRY

  TRY {
   pSim->step (0, stopT, 9, 2, IT_X);
   cout << " send info to Sim " << endl;
  } CATCHANY {
    cout << "Exception is SRVoffer \n";
    cout << IT_X;
    exit (-1);
  } ENDTRY

  }
```

B.5 SERVERMain.C

```
// The executable file generated from this code should be registered
// (under the name 'SERVER').

#include <stream.h>
#include "SERVER_i.h"
#include "SRVcontract_i.h"
#include <stdlib.h>

#include <IR.h>
#include "utils.h"
#include "echo.h"

void fatal (char* s, CORBA::Environment&e=CORBA::default_environment) {
    cout << s;
    if (e)
              cout << e;
    cout << endl;
    exit(-1);
}


int main(int argc, char **argv) {

  Repository* rep;
  InterfaceDef* contractInterface;
  InterfaceDef* my_interface;
  AttributeDef* contractAttributes;

  InterfaceDef::FullInterfaceDescription BD;

  _IDL_SEQUENCE_RepositoryId base_interfaces;
  Contained *contd;

  Identifier namelist[20];

  CI CTemp(20);

  CORBA::typeCode zz;
  CORBA::any anyzez;
  short sh;
  float fl;
  char *st;
  char* c;

  float QoSmax, pricemin;

  unsigned long ll = 0;
  unsigned long ElNo = 0;


void print_base_interfaces (ostream &o, Repository *rep,
                                                        InterfaceDef
*my_interface);

_IDL_SEQUENCE_RepositoryId get_base_interfaces (InterfaceDef *my_interface);
```

```
  if (argc < 3) {
     cout << "usage: "<<argv[0] << " <hostname> <interface_type> " << endl;
     exit (-1);
   }

// Call *impl_is_ready* initially to solve the persistent storage problem

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("SERVER3",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // bind now to the Interface Repository

  TRY {
   rep = Repository::_bind(":IR", argv[1], IT_X);
   }
  CATCHANY {
   cout << "Repository bind failed: " <<  IT_X << endl;;
   }
  ENDTRY

// Find all interfaces in the IR"

  rep->loadIDLAll ();

  struct _IDL_SEQUENCE_Contained myRepositoryContents;

// lookup initially for InterfaceDef specified by SERVER's argv[2]

  TRY{
    myRepositoryContents = rep->lookup_name (argv[2], -1, "InterfaceDef", 0);
  }
  CATCHANY{
   cout << "Rep-> contents() failed: " <<  IT_X << endl;
  }
  ENDTRY

// get info about the interface, and AttributeDescription ultimately

  my_interface = InterfaceDef::_narrow (myRepositoryContents[0]);

  base_interfaces = get_base_interfaces (my_interface);


  if (base_interfaces._length > 0)
     cout << "contract type: " << argv[2] << " has " <<
       base_interfaces._length << " base interface(s): " << endl << endl;

     for (unsigned long i = 0; i < base_interfaces._length; i++)
              {
              RepositoryId rep_id = base_interfaces[i];
```

```
              TRY {
                contd = rep->lookup_id(rep_id, IT_X);
                if (!IT_X)
                  {
                  cout << contd->name();
                  namelist[ll] = new char[20];
                  namelist[ll] = contd->name();
                  ll++;
                  }
              }
              CATCHANY {
                fatal ("Repository:lookup_id failed: ", IT_X);
              }
              ENDTRY
                if (i < (base_interfaces._length - 1))
                  cout << ", ";
     }

  namelist[ll] = new char[20];
  namelist[ll] = argv[2];

  cout << endl;

  for (unsigned long kk = 0; kk <=ll; kk++){
    TRY{
     myRepositoryContents=rep->lookup_name(namelist[kk],-
1,"InterfaceDef",0);
    }
    CATCHANY{
      cout << "Rep-> contents() failed: " <<  IT_X << endl;
    }
    ENDTRY

    my_interface = InterfaceDef::_narrow (myRepositoryContents[0]);
    BD = my_interface -> describe_interface ();
    cout << endl << "Interface type *" << BD.name << "*  has " <<
    BD.attributes._length << " contract elements:" << endl << endl;

    CTemp._length += BD.attributes._length;

    for (unsigned long m = 0; m < BD.attributes._length; m++)
     {
     zz = BD.attributes[m].type;
     CORBA::TCkind CEkind = zz.kind();
     cout << endl << "element No. " << m << " is of type " ;
     switch (CEkind) {
       case CORBA::tk_short:
               c = "short";
               break;
       case CORBA::tk_float:
         c = "float";
               break;
       case CORBA::tk_string:
         c = "string";
               break;
       }
     cout << c << " with name " << BD.attributes[m].name << endl;
```

```
     cout << "Please enter the value for this element: " << endl;

     switch (CEkind) {
       case CORBA::tk_short:
         cin >> sh;
               CTemp._buffer[ElNo].CEvalue << sh;
               break;
       case CORBA::tk_float:
               cin >> fl;
               CTemp._buffer[ElNo].CEvalue << fl;
               break;
       case CORBA::tk_string:
               st = new char[20];
         cin >> st;
               CORBA::any anyzez (CORBA::TC_string, &st);
               CTemp._buffer[ElNo].CEvalue = anyzez;
               break;

       }
       CTemp._buffer[ElNo].CE = BD.attributes[m] ;
       ElNo++;
   } // for m
  } // for kk

  cout << endl << endl;
  cout << "The contract instance derived has the following form:" << endl;
  printCI(CTemp);

// The local sequence CTemp is now filled with:
//    a) description about contract element TYPES
//       (from the IR)
//    b) the corresponding initial VALUES
//       (entered by a human during SERVER startup)

// constractor for the SERVER object

  cout << "Please enter value for maximum QoS that can be provided: "
    << endl;
  cin >> QoSmax;
  cout << "Please enter value for minimum price that can be accepted: "
    << endl;
  cin >> pricemin;
  cout << " QoSmax = " << QoSmax <<  "  pricemin =" <<  pricemin << endl;

  SERVER_i myContract (CTemp, 10.0, argv[2], QoSmax, pricemin);

  cout << endl << endl;
  cout << "Initial SERVER's Contract Instance created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:

CORBA::Orbix.impl_is_ready("SERVER3",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
```

```
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "SERVER exiting" << endl;

  return 0;

}
```

B.6 CV_i.h

```
#ifndef CV_ih
#define CV_ih

#include "CV.hh"
#include <IR.h>


class CV_i : public virtual CVBOAImpl {
   Repository* m_rep;

   BadServers* pBadServers;

public:
              // constructor
              CV_i(char* host, unsigned long Init);

              virtual void checkValidity (const CI& x, const char *
contract_type,
short& answer, CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void checkReputation (const char * Server_name,
short& Rep_answer, CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void addServer (const char * Server_name,
CORBA::Environment &IT_env=CORBA::default_environment);

};


#endif
```

B.7 CV_i.C

```
#include "CV_i.h"
#include <stream.h>
#include <stdlib.h>

#include <unistd.h>
#include <IR.h>
#include "echo.h"

void fatal (char* s, CORBA::Environment&e=CORBA::default_environment) {
    cout << s;
    if (e)
              cout << e;
    cout << endl;
    exit(-1);
}



CV_i::CV_i (char* host, unsigned long Init){

Repository* repz;

  // attempt to bind to the Interface Repository

  TRY {
   repz = Repository::_bind(":IR", host, IT_X);
   }
  CATCHANY {
   cout << "Repository bind failed: " <<  IT_X << endl;
   }
  ENDTRY

  m_rep = repz;
cout << "successfully bound to the IR " << endl;

  pBadServers = new BadServers(Init);
  pBadServers->_length = 0;
  pBadServers->_maximum = Init;
}



void CV_i:: checkValidity (const CI& x,  const char* contract_type,
                                              short& answer,
CORBA::Environment &IT_env) {

  Repository* rep;
  rep = m_rep;

  CI CTemp(20);

  InterfaceDef* contractInterface;
  InterfaceDef* my_interface;
  AttributeDef* contractAttributes;
  InterfaceDef::FullInterfaceDescription BD;
```

```
  struct _IDL_SEQUENCE_Contained myRepositoryContents;

  _IDL_SEQUENCE_RepositoryId base_interfaces;
  Contained *contd;

  Identifier namelist[20];

  cout << "Entered *checkValidity* operation " << endl;

void print_base_interfaces (ostream &o, Repository *rep,
                                                    InterfaceDef
*my_interface);

_IDL_SEQUENCE_RepositoryId get_base_interfaces (InterfaceDef *my_interface);

// load the IR, which includes CR.idl (description of contarct templates)

  rep->loadIDLAll ();

  char* temp = new char[50];
  strcpy(temp, contract_type);

  TRY{
    myRepositoryContents = rep -> lookup_name (temp, -1, "InterfaceDef", 0);
  }
  CATCHANY{
   cout << "Rep-> contents() failed: " << IT_X;
  }
  ENDTRY

  cout << "contract_type is: " << contract_type << endl;

  my_interface = InterfaceDef::_narrow (myRepositoryContents[0]);

  base_interfaces = get_base_interfaces (my_interface);

  unsigned long ll = 0;
  unsigned long ElNo = 0;

  if (base_interfaces._length > 0)
     cout << "contract type: " << temp << " has "
       << base_interfaces._length << " base interface(s): " << endl;

     for (unsigned long i = 0; i < base_interfaces._length; i++)
               {
               RepositoryId rep_id = base_interfaces[i];
               TRY {
                 contd = rep->lookup_id(rep_id, IT_X);
                 if (!IT_X)
                   {
                   cout << contd->name();
                   namelist[ll] = new char[20];
                   namelist[ll] = contd->name();
                   ll++;
                   }
               }
               CATCHANY {
                 fatal ("Repository:lookup_id failed: ", IT_X);
```

```
              }
              ENDTRY
                if (i < (base_interfaces._length - 1))
                  cout << ", ";
      }

  namelist[ll] = new char[20];
  namelist[ll] = temp;

  cout << endl;

  for (unsigned long kk = 0; kk <=ll; kk++){
    TRY{
     myRepositoryContents=rep->lookup_name(namelist[kk],-
1,"InterfaceDef",0);
    }
    CATCHANY{
      cout << "Rep-> contents() failed: " <<  IT_X << endl;
    }
    ENDTRY

    my_interface = InterfaceDef::_narrow (myRepositoryContents[0]);
    BD = my_interface -> describe_interface ();
    cout << endl << "Interface type *" << BD.name << "*  has " <<
    BD.attributes._length << " contract elements:" << endl << endl;

    CTemp._length += BD.attributes._length;

    for (unsigned long m = 0; m < BD.attributes._length; m++)
      CTemp._buffer[ElNo++].CE = BD.attributes[m] ;

  }

  if (x._length == CTemp._length)
    {
      answer = 1;
      for (unsigned long i; i < x._length; i++)
              {
                if ( CTemp[i].CE.name != x[i].CE.name)
                  answer = 0;
                if ( CTemp[i].CE.type != x[i].CE.type) // etc ...
                  answer = 0;
              }
    }
  else
      answer = 0;
  cout << "Exit *checkValidity* operation " << endl;
}

void CV_i:: checkReputation (const char * Server_name, short& Rep_answer,
                                                CORBA::Environment &IT_env) {
short temp = 1;
unsigned long i;
  cout << "Enter *checkReputation* operation " << endl;

i=0;
while (i++ < pBadServers->_length && (temp == 1))
  if (strcmp (Server_name, pBadServers->_buffer[i]) == 0)
```

```
    temp = 0;

Rep_answer = temp;
cout << "Server_name " << Server_name << " answer = " << Rep_answer;
  cout << "Exit *checkReputation* operation " << endl;
}

void CV_i :: addServer (const char * Server_name, CORBA::Environment &IT_env)
{
  cout << "Enter *addServer* operation " << endl;
pBadServers->_buffer[pBadServers->_length] = new char[strlen(Server_name)];
strcpy( pBadServers->_buffer[pBadServers->_length], Server_name);
pBadServers->_length++;

if (pBadServers->_length >pBadServers->_maximum ){
              cout << "Initial number exceeded !! " << endl;
              pBadServers->_maximum = pBadServers->_length;
              }

  cout << "Exit *addServer* operation " << endl;
}
```

B.8 CVMain.C

```
// The executable file generated from this code should be registered
// (under the name 'CV').


#include <stream.h>
#include "CV_i.h"
#include <stdlib.h>
#include <IR.h>

int main(int argc, char **argv) {

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname>" << endl;
     exit (-1);
   }

// Call *impl_is_ready* initially to solve the persistent storage problem

  TRY {
    CORBA::Orbix.impl_is_ready("CV2",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

// constractor for the CV object

  CV_i CVcomponent (argv[1], 50);

  cout << "CV Component created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("CV2",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "server exiting" << endl;

  return 0;

}
```

B.9 Notary_i.h

```
#ifndef Notary_ih
#define Notary_ih

#include "Notary.hh"

class Notary_i : public virtual NotaryBOAImpl {
        CIList m_CIList;
        CIList* pCIList;
              unsigned long m_Last_used;
              unsigned long m_Max;
public:
              // constructor

              Notary_i(unsigned long Init);

    virtual CIList NL (CORBA::Environment
&IT_env=CORBA::default_environment);

              virtual void store (const CI& x, unsigned long& No,
 CORBA::Environment &IT_env=CORBA::default_environment);

              virtual void retrieve (unsigned long ContractNo, CI& Instance,
CORBA::Environment &IT_env=CORBA::default_environment);
};


#endif
```

B.10 Notary_i.C

```
#include "Notary_i.h"

#include <stream.h>
#include <stdlib.h>

CIList Notary_i :: NL (CORBA::Environment &IT_env) {
return  m_CIList;
}

Notary_i::Notary_i (unsigned long Init){

pCIList = new CIList (Init);
pCIList->_length = 0;
pCIList->_maximum = Init;

} // c-tor


void Notary_i:: store (const CI& x, unsigned long& No,
                                    CORBA::Environment &IT_env) {

cout << "Entered *store* operation " << endl;
pCIList->_buffer[pCIList->_length] = x;

No = pCIList->_length;
(pCIList->_length)++;

if (pCIList->_length >pCIList->_maximum ){
              cout << "Initial number exceeded !! " << endl;
              pCIList->_maximum = pCIList->_length;
              }
cout << "Exit *store* operation " << endl;

} // store

void Notary_i:: retrieve (unsigned long ContractNo, CI&
Instance,                                        CORBA::Environment &IT_env) {

cout << "Entered *retrieve* operation " << endl;

Instance = pCIList->_buffer[ContractNo];

cout << "Exit *retrieve* operation " << endl;

} // retrieve
```

B.11 NotaryMain.C

```
// The executable file generated from this code should be registered
// (under the name 'Notary').


#include <stream.h>
#include "Notary_i.h"
#include <stdlib.h>

int main(int argc, char **argv) {

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname>" << endl;
     exit (-1);
   }

// constractor for the Notary object

  TRY {
    CORBA::Orbix.impl_is_ready("Notary",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY


  Notary_i MyNotary (100);

  cout << "Notary Component created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
   CORBA::Orbix.impl_is_ready("Notary",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "server exiting" << endl;

  return 0;

}
```

B.12 CN_i.h

```
#ifndef CN_ih
#define CN_ih

#include "SERVER.hh"
#include "CN.hh"

class CN_i: public virtual CNBOAImpl {
  SERVER* m_pS;

public:

// constructor
  CN_i (char* host);

  virtual void BestDeal (CI& Best, CORBA::Environment
&IT_env=CORBA::default_environment);
};

#endif
```

B.13 CN_i.C

```
#include "CN_i.h"
#include <stream.h>
#include <stdlib.h>
#include "utils.h"
#include <unistd.h>


CN_i::CN_i (char* host){

SERVER* pS;

  TRY {
   pS = SERVER::_bind(":SERVER3", host, IT_X);
   }
  CATCHANY {
   cout << "Server bind failed: " <<  IT_X << endl;
   }
  ENDTRY

  m_pS = pS;
cout << "successfully bound to the SERVER " << endl;
}


// CN works on behalf of the CLIENT. It would try to find the best deal for it!

void CN_i:: BestDeal (CI& Best,  CORBA::Environment &IT_env){

SERVER* pS;

  CI a;
  CI counteroffer;
  char* contract_type;
  float QoSfactor, pricefactor;
  float QoSmax, pricemin;
  pS = m_pS;

  short Saccept = 0;

// get info from the SERVER

  cout << endl;
  cout << "Entered BestDeal procedure " << endl << endl;

  contract_type = new char[50];
  TRY {
   pS->SRVoffer (a, contract_type, IT_X);
   cout << "contract_type is " << contract_type << endl;
  } CATCHANY {
    cout << "Exception is SRVoffer \n";
    cout << IT_X;
    exit(-1);
  } ENDTRY

  cout << " Initial offer is of the following form " << endl;
```

```
  printCI(a);

  TRY {
   pS->SpecialInfo ("CN", QoSfactor, pricefactor,IT_X);
   cout << "QoSfactor is: " << QoSfactor << "  pricefactor is: " << pricefactor
<< endl;
  } CATCHANY {
    cout << "Exception is SpecialInfo \n";
    cout << IT_X;
    exit (-1);
  } ENDTRY

  QoSmax = *( (float*) a[2].CEvalue._value) * QoSfactor;
  pricemin = *( (float*) a[3].CEvalue._value) * pricefactor;

  cout << "SERVER can provide QoSmax of: " << QoSmax << " and will accept
minimum of: " << pricemin <<  " $" << endl;

  counteroffer = a;
  counteroffer[2].CEvalue << QoSmax;
  counteroffer[3].CEvalue << pricemin;

  TRY {
    cout << "CN is sending counteroffer now ... " << endl;
    pS->CLoffer (0, counteroffer, IT_X);
    pS->SRVresponse (Saccept, IT_X);

    if (Saccept == 1){
      cout << "Server accepted counteroffer " << endl;
      Best = counteroffer;
      }
    else
      cout << "Server has not accepted the counteroffer - left to CLIENT to
proceed" << endl;
  } CATCHANY {
    cout << "Exception in CLTResponse\n";
    cout << IT_X;
    exit(-1);
  } ENDTRY
  cout << "Exit BestDeal procedure " << endl << endl;

}
```

B.14 CNMain.C

```
// The executable file generated from this code should be registered
// (under the name 'CN').


#include <stream.h>
#include "CN_i.h"
#include <stdlib.h>
#include <IR.h>

int main(int argc, char **argv) {

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname>" << endl;
     exit (-1);
   }

// Call *impl_is_ready* initially to solve the persistent storage problem

  TRY {
    CORBA::Orbix.impl_is_ready("CN",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

// constractor for the CN object

  CN_i CNcomponent (argv[1]);

  cout << "CN Component created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("CN",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "server exiting" << endl;

  return 0;
```

B.15 CM_i.h

```
#ifndef CM_ih
#define CM_ih

#include "CM.hh"
#include "CE.hh"
#include "Notary.hh"

class CM_i : public virtual CMBOAImpl  {

  CE* m_pCE;         // needed to use CE
  Notary* m_pNotary; // needed to use Notary

  CMlist m_CMlist;
  CMlist* pCMlist;
  unsigned long m_CM_Last_used;
  unsigned long m_CM_Max;

public:
  //constructor
  CM_i(unsigned long Init,  char* host);

    virtual CMlist CML (CORBA::Environment
&IT_env=CORBA::default_environment);

              virtual void request (unsigned long CoNo, const char * host,
const char * nameS, float QoSdegrad, const char * what_to_do, unsigned long
stop,  CORBA::Environment &IT_env=CORBA::default_environment);
};

#endif
```

B.16 CM_i.C

```
#include "CM_i.h"

#include <stream.h>
#include <stdlib.h>

#include "SERVER.hh"

#include "utils.h"
#include <unistd.h>

#include <time.h>


CMlist CM_i ::  CML (CORBA::Environment &IT_env) {
return m_CMlist;
}

CM_i::CM_i(unsigned long Init, char* host){

pCMlist = new CMlist (Init);
pCMlist->_length = 0;
pCMlist->_maximum = Init;
}


void CM_i :: request (unsigned long CoNo, const char * host, const char *
nameS, float QoSdegrad, const char * what_to_do, unsigned long stop,
CORBA::Environment &IT_env) {

  Notary* pN;
  CE* pCE;
  SERVER* pSERVER;
  struct tm* T;

  float zezread, zezwrite, zezfirst, factor;
  CI out;

  char warn[50];
  strcpy (warn, "QoS degradation has been noted!");

  cout << endl;
  cout << "Entered *request* operation " << endl << endl;
  cout << "Contract number to be monitored is : " << CoNo << endl;
  // bind to the Notary on the node argv[1] first

  TRY {
    pN = Notary::_bind(":Notary", host, IT_X);
    cout << "CM now bound to Notary to obtain Contract Instance " << endl;
  } CATCHANY {
    cerr << "Bind to Notary failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(1);
  } ENDTRY

    m_pNotary = pN;
```

```
  char temp[10];
  strcpy (temp, what_to_do);

  time_t tiempo;
  tiempo = time(NULL);

  cout << ctime(&tiempo);

  TRY {
    m_pNotary->retrieve(CoNo, out, IT_X );
    cout << endl;
    cout << "Contact Instance obtained from Notary is: " << endl;
    printCI (out);
  } CATCHANY {
    cerr << "Invoke retrieve failed" << IT_X << endl;
    exit(-1);
  } ENDTRY

  TRY {
    pSERVER = SERVER::_bind(":SERVER3", host, IT_X);
  } CATCHANY {
    cerr << "Bind to SERVER from CM failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(-1);
  } ENDTRY

  tiempo = time(NULL);
  T = localtime(&tiempo);

  cout << " T->tm_min = " << T->tm_min << " T->tm_sec = " << T->tm_sec << "
stop = " << stop << endl;


while ( (T->tm_min < stop) ) {

   unsigned long ClientNo = 0; // Client No 1
    for (unsigned long  m=0; m < 10; m++){
      TRY { // this just for checking and printing
              pSERVER->getQoS(ClientNo, zezread, IT_X);

              if (zezread < QoSdegrad )
                {
                  tiempo = time(NULL);
                  cout << ctime(&tiempo);

                  if ( strcmp(temp,"record") == 0){
                    pCMlist->_buffer[pCMlist->_length].CoNo = CoNo;
//                  pCMlist->_buffer[pCMlist->_length].host = host;
                    pCMlist->_buffer[pCMlist->_length].QoSvalue = zezread;
                  cout << " pCMlist->_length =" <<
                    pCMlist->_length << "  value= "
                            << pCMlist->_buffer[pCMlist->_length].QoSvalue
<< endl;
                  (pCMlist->_length)++;

                  if (pCMlist->_length > pCMlist->_maximum ){
                    cout << "Initial number exceeded !! " << endl;
```

```
                    pCMlist->_maximum = pCMlist->_length;
                  }
                  } //if strcmp
                  else {
                    cout << "will send message to CE " << endl;
                    TRY {
                            pCE = CE::_bind(":CE", host, IT_X);
//                          cout << "bound to CE" << endl;
                    } CATCHANY {
                            cerr << "Bind to CE failed" << endl;
                            cerr << "Unexpected exception " << IT_X << endl;
                            exit(1);
                    } ENDTRY

                            m_pCE = pCE;

                    TRY {
                    m_pCE->act_on_server(warn, host, nameS, IT_X);
                    } CATCHANY {
                            cerr << "Invoke act_on_server failed" << IT_X <<
endl;
                            exit(-1);
                    } ENDTRY

                    } // else
                }

      } CATCHANY {
              cout << "Exception in access to currentQoS \n" <<  IT_X;
              exit(-1);
      } ENDTRY
      sleep (2);     // determine time step

    } // for m

  tiempo = time(NULL);
  T = localtime(&tiempo);
  cout << " T->tm_min = " << T->tm_min << " T->tm_sec = " << T->tm_sec << "
stop = " << stop << endl;


} // while (T->tm_min < stop)

  cout << "Monitor stops this monitoring " << endl << endl;

  cout << "Exit *request* operation " << endl << endl;
}
```

B.17 CMMain.C

```
// The executable file generated from this code should be registered
// (under the name 'CM').


#include <stream.h>
#include "CM_i.h"
#include <stdlib.h>
#include "Notary.hh"
#include "CE.hh"

int main(int argc, char **argv) {
  Notary* pNotary;
  CE* pCE;

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname>" << endl;
     exit (-1);
   }

// constractor for the CM object

  TRY {
    CORBA::Orbix.impl_is_ready("CM",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY


  CM_i MyCM (100, argv[1]);

  cout << "CM Component created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("CM",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "server exiting" << endl;

  return 0;

}
```

B.18 CE_i.h

```
#ifndef CE_ih
#define CE_ih

#include "CE.hh"
#include "SERVER.hh"

class CE_i : public virtual CEBOAImpl  {

  CElist m_CElist;
  CElist* pCElist;
  unsigned long m_CE_Last_used;
  unsigned long m_CE_Max;

public:
  //constructor

  CE_i(unsigned long Init);

    virtual CElist CEL (CORBA::Environment
&IT_env=CORBA::default_environment);

    virtual void act_on_server (const char * message, const char * host,
const char * nameS, CORBA::Environment &IT_env=CORBA::default_environment);
};

#endif
```

B.19 CE_i.C

```
#include "CE_i.h"
#include "CV_i.h"

#include <stream.h>
#include <stdlib.h>

CElist CE_i ::  CEL (CORBA::Environment &IT_env) {
return m_CElist;
}

CE_i::CE_i(unsigned long Init){

pCElist = new CElist (Init);
pCElist->_length = 0;
pCElist->_maximum = 1;

}


void CE_i:: act_on_server (const char * message, const char * host,
const char * nameS, CORBA::Environment &IT_env) {

  float zezread, zezwrite, zezfirst, factor;

  SERVER* pSERVER;
  CV* pCV;

  TRY {
    pSERVER = SERVER::_bind(":SERVER3", host, IT_X);

  } CATCHANY {
    cerr << "Bind to SERVER failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(1);
  } ENDTRY

  TRY {
    pSERVER->SRVenforce(message, IT_X);
    cout << "CE sends message to SERVER .... " << endl;
  } CATCHANY {
    cerr << "Bind to SERVER from CM failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(-1);
  } ENDTRY



  TRY {
    pCV = CV::_bind(":CV2", host, IT_X);
    cout << "bound to CV " << endl;
  } CATCHANY {
    cerr << "Bind to CV failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(1);
  } ENDTRY
```

```
  TRY {
    pCV->addServer(nameS,  IT_X);
    cout << " add SERVER name to CV  " << endl;
  } CATCHANY {
    cerr << " addServer  failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(-1);
  } ENDTRY


}
```

B.20 CEMain.C

```
// The executable file generated from this code should be registered
// (under the name 'CE').

#include <stream.h>
#include <stdlib.h>
#include "CE_i.h"

int main(int argc, char **argv) {

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname>" << endl;
     exit (-1);
   }

// Call *impl_is_ready* initially to solve the persistent storage problem

  TRY {
    CORBA::Orbix.impl_is_ready("CE",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

// constractor for the CE object

  CE_i myCE (100);

  cout << "CE Component created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("CE",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // impl_is_ready() returns only when Orbix times-out an idle server
  // (or an error occurs).

  cout << "server exiting" << endl;

  return 0;

}
```

B.21 Sim_i.h

```
#ifndef Sim_ih
#define Sim_ih

#include "Sim.hh"
#include "SERVER.hh"


class Sim_i : public virtual SimBOAImpl {

  QoS_sim* pQoS_sim;

public:

        Sim_i (unsigned long Init);

              virtual void step (unsigned long ClientNo, unsigned long stop,
short change, short drop, CORBA::Environment
&IT_env=CORBA::default_environment);


};


#endif
```

B.22 Sim_i.C

```
#include "Sim_i.h"
#include "SERVER_i.h"

#include <time.h>
#include <stream.h>
#include <stdlib.h>
#include <unistd.h>

Sim_i::Sim_i ( unsigned long Init) {

// Create a pointer to a sequence with Clients' QoS data

pQoS_sim = new QoS_sim (Init);
pQoS_sim->_length = 0;   // always points to the last used !!
pQoS_sim->_maximum = Init;

} // end constructor


void Sim_i:: step (unsigned long ClientNo, unsigned long stop,
short change, short drop, CORBA::Environment &IT_env)  {

SERVER* pSERVER;
time_t tiempo;
struct tm* T;

float factor;

// Create a QoS value for new client:

 cout << "Start producing QoS now ... " << endl;

 cout << " try to bind to the SERVER .... " << endl;

  TRY {
    pSERVER = SERVER ::_bind(":SERVER3", "foxtail.dstc.edu.au", IT_X);
  } CATCHANY {
    cerr << "Bind to SIM  object failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(1);
  } ENDTRY

  float readQoS, writeQoS, firstQoS;

  TRY {
   pSERVER->getQoS (ClientNo, firstQoS, IT_X);
   cout << "firstQoS = " << firstQoS << endl;
  } CATCHANY {
    cout << "Exception is SRVoffer \n"  << IT_X;
    exit (-1);
  } ENDTRY

  tiempo = time(NULL);
  T = localtime(&tiempo);
```

```
  cout << " T->tm_min = " << T->tm_min << " T->tm_sec = "
    << T->tm_sec << " stop = " << stop << endl;


while ( (T->tm_min < stop) ) {
  cout << "QoS delivery to be varied as follows:  " << endl;

  for (unsigned long  m=0; m < 10; m++){

    sleep (5);     // determines time step
    if (m < change)
      factor = 0;
    else
      factor = drop;
    writeQoS = -factor + firstQoS;

   TRY {
    pSERVER->setQoS (ClientNo, writeQoS, IT_X);
    cout << "writeQoS = " << writeQoS << endl;
   } CATCHANY {
     cout << "Exception is setQoS \n"  << IT_X;
     exit (-1);
   } ENDTRY

   }

  tiempo = time(NULL);
  T = localtime(&tiempo);
  cout << " T->tm_min = " << T->tm_min << " T->tm_sec = "
    << T->tm_sec << " stop = " << stop << endl;


} // while (T->tm_min < stop)

 cout << "Will stop simulating service executiion now !!! " << endl;

}
```

B.23 SimMain.C

```
// A SIMULATOR for the SERVER example.

#include <stream.h>

#include "Sim_i.h"

#include <stdlib.h>
#include <unistd.h>

main (int argc, char **argv) {

  SERVER* pSERVER;   // pointer to the SERVER object that will be used.

  short time_of_change, drop;

  if (argc < 2) {
     cout << "usage: " << argv[0] << " <hostname> " << endl;
     exit (-1);
   }

  cin >> time_of_change;
  cin >> drop;

// Call *impl_is_ready* initially to solve the persistent storage problem

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("SIM",0,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
  }
  ENDTRY

  // bind to the SERVER on the node argv[1] first
  TRY {
    pSERVER = SERVER::_bind(":SERVER3", argv[1], IT_X);
  } CATCHANY {
    cerr << "Bind to SERVER failed" << endl;
    cerr << "Unexpected exception " << IT_X << endl;
    exit(-1);
  } ENDTRY

  Sim_i mySim (10);

  cout << "Simulator object for SERVER's QoS variation  created !" << endl;

  TRY {
    // tell Orbix that we have completed the server's initialisation:
    CORBA::Orbix.impl_is_ready("SIM",CORBA::Orbix.INFINITE_TIMEOUT,IT_X);
  }
  CATCHANY {
    // an error occured calling impl_is_ready() - output the error.
    cout << IT_X;
```

```
  }
  ENDTRY

  return 0;

} // main
```

B.24 CLIENT.C

```
// A client for the SERVER example.

//
// Assumptions:
//
// 1. CLIENT knows
//            a) type of service wanted (e.g. trading service): found
previously
//            b) server ID and its characteristics
// 2. SERVER is initially trusted (during negotiation) but:
//            a) CLIENT can subsequently check contract validity
//            b) this would require access to ORBIX (CORBA) Interface
Repository(IR)
//     c) the validity checking will be performed by Contract Validator (CV)
// 3. Once the contract instance is stored in the Notary object (CLIENT will
//    do this), CLIENT will notify the Contract Monitor (CM) object to start
//    monitoring SERVER's QoS
// 4. Enforcing starts with sending (the first) warning message to the SERVER,
//    recording invalid SERVER's behaviour and indicating this to the CV
//


#include <stream.h>
#include <time.h>

#include "SERVER.hh"

#include "CV.hh"
#include "Notary.hh"
#include "CM.hh"
#include "CE.hh"
#include "CN.hh"

#include <unistd.h>

#include <stdlib.h>

#include "utils.h"


main (int argc, char **argv) {

  time_t tiempo;
  struct tm* T;
  unsigned long startT;
  unsigned long stopT;

  unsigned long start_time, stop_time;


  SERVER* pSERVER;     // pointer to the SERVER object that will be used.
  CV* pCV;             // pointer to the CV object that will be used.
  Notary* pNotary;
  CM* pCM;             // pointer to the CM object that will be used.
  CE* pCE;             // pointer to the CE object that will be used.
```

```
CN* pCN;             // pointer to the CN object that will be used.

float QoSvalue, price;
float price_limit; // that CLIENT wants to pay
float QoS_limit;   // that CLIENT accepts in terms of QoS value
short Caccept;
short Saccept = 0;
short valid, Rep_answer;
char key[10], keyNot[10], useCN[10];

unsigned long CNo;

char* contract_type;

CI a;
CI counteroffer;
CI to_go;
CI out;

CMlist CM_II;

if (argc < 2) {
   cout << "usage: " << argv[0] << " <hostname> " << endl;
   exit (-1);
 }

char* ClientType;
ClientType = new char[strlen (argv[2]) +1];
strcpy (ClientType, argv[2]);

if (strcmp (ClientType, "easy_going") == 0 )
    Caccept = 1;
else if (strcmp (ClientType, "negotiator") == 0 )
    Caccept = 0;
else
    {cout << "wrong CLIENT's characteristic type " << endl; exit (-1);}

cout << "Binding to the SERVER ..." << endl;

TRY {
  pSERVER = SERVER::_bind(":SERVER3", argv[1], IT_X);
  cout << "Client bound to server "  << endl;
} CATCHANY {
  cerr << "Bind to SERVER object failed" << endl;
  cerr << "Unexpected exception " << IT_X << endl;
  exit(1);
} ENDTRY



cout << endl;
cout << "Next step is to check SERVER's initial offer " << endl;
cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
cin >> key;
if ( strcmp(key, "y") != 0) {
  cout << "Will exit now " << endl;
  return 0;
}
```

```
  cout << "Check SERVER's initial offer now ... " << endl << endl;
  contract_type = new char[50];
  TRY {
   pSERVER->SRVoffer (a, contract_type, IT_X);
   to_go = a;
  } CATCHANY {
    cout << "Exception is SRVoffer \n";
    cout << IT_X;
    return -1;
  } ENDTRY

  cout << "Contract type is: " << contract_type << endl;
  cout << "Initial offer is of the following form: " << endl;
  printCI(a);

  if  ( Caccept == 1 ) {
    TRY {
     pSERVER->CLoffer(Caccept, counteroffer, IT_X);
    } CATCHANY {
     cout << "Exception is CLoffer \n" << IT_X;
     return -1;
    } ENDTRY
   }
  else {
   cout << endl;
   cout << "Next step is Negotiation (if client is negotiator) " << endl;
   cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
   cin >> key;

  if ( strcmp(key, "y") == 0) { // Negotiation
   cout << "CLIENT is: " << ClientType << endl;
    cout << "Is CN to be used (for price)? " << endl;
    cin >> useCN;
   if ( strcmp(useCN, "y") != 0) { // Negotiation betwen client and server
    cout << "Negotiation between CLIENT and SERVER starts now " << endl <<
endl;
     while (!Saccept) {
      cout << "Enter Client's counteroffer " << endl;
      cout << "Enter a number for minimum acceptable QoS: " << endl;
      cin >> QoS_limit;
      cout << "Enter a number for maximum price: " << endl;
      cin >> price_limit;

      CI to_go (a._length);
      to_go._length = a._length;

      to_go = a;
      to_go[2].CEvalue << QoS_limit;
      to_go[3].CEvalue << price_limit;

      counteroffer = to_go;

      TRY {
              cout << "Client is sending counteroffer now ... " << endl;
              pSERVER->CLoffer(Caccept, counteroffer, IT_X);
              cout << "Client waits for SERVER's decesion ..." << endl;
              sleep (3);
```

```
              pSERVER->SRVresponse(Saccept, IT_X);

              if (Saccept == 1)
                cout << "Server accepted counteroffer " << endl;
              else
                cout << "Server has not accepted the counteroffer - left to
CLIENT to proceed" << endl;

      } CATCHANY {
              cout << "Exception in CLTResponse\n " <<  IT_X;
              return -1;
      } ENDTRY
     } // while
 } // if useCN

    else
      {
              cout << "invoke CN here " << endl;
              cout << "Binding to the CN ..." << endl;
              TRY {
                pCN = CN::_bind(":CN", argv[1], IT_X);
                cout << "Client bound to CN "  << endl;
              } CATCHANY {
                cerr << "Bind to object failed" << endl;
                cerr << "Unexpected exception " << IT_X << endl;
                exit(1);
              } ENDTRY

              TRY{
                pCN->BestDeal (out, IT_X);
                cout << "Best deal found through Negotiator is: " << endl;
                printCI(out);
              }
              CATCHANY{
                cerr << "Exception in PCV->checkV \n " <<  IT_X << endl;
                return -1;
              }ENDTRY
      }
  cout << "Negotiation completed ! " << endl;
 } // Negotiaton
} // else accept

  cout << endl;
  cout << "Next step is Validation  " << endl;
  cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
  cin >> key;
  if ( strcmp(key, "y") == 0) {
    cout << "Binding to the Contract Validator ..." << endl;
    TRY {
      pCV = CV::_bind(":CV2", argv[1], IT_X);
      cout << "sucessfully bound to CV " << endl;
    } CATCHANY {
      cerr << "Bind to object failed" << endl;
      cerr << "Unexpected exception " << IT_X << endl;
      exit(1);
    } ENDTRY

    TRY{
```

```
      pCV->checkValidity (to_go, contract_type, valid, IT_X);
    }
    CATCHANY{
      cerr << "Exception in PCV->checkV \n " <<  IT_X << endl;
      return -1;
    }ENDTRY

    cout << "Contract is valid - check SERVER's reputation now " << endl;
    TRY{
      pCV->checkReputation ("SERVER3", Rep_answer, IT_X);
    }
    CATCHANY{
      cerr << "Exception in PCV->checkV \n " <<  IT_X << endl;
      return -1;
    }ENDTRY

    if (Rep_answer == 0)
      cout << "SERVER does not have a good reputation - not worth pursuing
this further !!! "  << endl;

    if (valid !=1 || Rep_answer == 0) {
      cout << " Contract not valid ... will exit now " << endl;
      return 0;
    }
    cout << "Contract is valid and SERVER is OK (reputable) " << endl;
  } // if Validation

  cout << endl;
  cout << "Next step is storing contract instance into Notary" << endl;
  cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
  cin >> keyNot;
  if ( strcmp(keyNot, "y") == 0) { // Notary
    cout << "Binding to the Notary now  ..." << endl;
    TRY {
      pNotary = Notary::_bind(":Notary", argv[1], IT_X);
    } CATCHANY {
      cerr << "Bind to Notary failed" << endl;
      cerr << "Unexpected exception " << IT_X << endl;
      exit(-1);
    } ENDTRY

    TRY {
      pNotary->store(to_go, CNo, IT_X );
      cout << "The stored contract instance has number: " << CNo << endl;
    } CATCHANY {
      cerr << "Invoke store failed" << endl;
      cerr << "Unexpected exception " << IT_X << endl;
      exit(-11);
    } ENDTRY

} // Notary

// start service here
  cout << endl;
  cout << "SERVER can now be invoked to start a service" << endl;
  cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
  cin >> key;
  if ( strcmp(key, "y") == 0) { // Invoke SERVER
```

```
    cout << "After what time [mins] do you want server to start" << endl;
    cin >> start_time;
    cout << "After what time [mins] do you want server to stop" << endl;
    cin >> stop_time;

   cout << "SERVER will be started in " << start_time << " mins  and stopped
after " << stop_time << "  mins" << endl;

    TRY {
      pSERVER->service(start_time, stop_time, argv[0], IT_X );
    } CATCHANY {
      cerr << "Invoke service failed" << IT_X << endl;
      exit(1);
    } ENDTRY

    tiempo = time(NULL);
    cout << "It is now " << ctime (&tiempo);
    cout << "Wait till SERVER starts ..." << endl;

    tiempo = time(NULL);
    T = localtime(&tiempo);

    if (T->tm_min >= (60-start_time))
      startT = T->tm_min -(60-start_time);
    else
      startT = T->tm_min + start_time;

    cout << "startT = " << startT << endl;

    if (startT > (60-stop_time) )
      stopT = startT -(60 - stop_time);
    else
      stopT = startT + stop_time;

    cout << "stopT = " <<  stopT << endl;
    cout << endl;

    tiempo = time(NULL);
    T = localtime(&tiempo);

    while (! (T->tm_min == startT) ) {
     cout << " T->tm_min = " << T->tm_min << " T->tm_sec = " << T->tm_sec <<
"      startT = " << startT << endl;
      sleep(10);
      tiempo = time(NULL);
      T = localtime(&tiempo);
    }

    cout << endl;
    cout << "Next step is reqesting actions of CM " << endl;
    cout << "DO YOU WANT TO CONTINUE WITH THIS ? [y/n] " << endl;
    cin >> key;
    if ( strcmp(key, "y") == 0) { // CM
      if ((strcmp(keyNot, "y") != 0)) {
               cout << "Contract Instance not previuosly stored -> Monitoring
not possible with CM " << endl;
               cout << "Will exit now " << endl;
               return 0;
```

```
    }

    cout << "Binding to the Contract Monitor  now  ..." << endl;
    TRY {
      pCM = CM::_bind(":CM", argv[1], IT_X);
      cout << "succesfully bound to CM " << endl;
    } CATCHANY {
      cerr << "Bind to CM failed - Unexpected exception " << IT_X << endl;
      exit(-1);
    } ENDTRY

    cout << " Pass arguments to Contract Monitor and fire it up! " << endl;

    cout << "What do you want CM to do? [record/invokeCE] " << endl;
    cin >> key;
   while ( ! ((strcmp(key,"record") == 0) || (strcmp(key,"invokeCE") == 0 )))
      {
               cout << "wrong entry: enter <record> or <invokeCE>" << endl;
               cin >> key;
      }

    TRY {
      pCM->request(CNo, argv[1], "SERVER3", 9 , key , stopT, IT_X );
    } CATCHANY {
      cerr << "Invoke request failed" << IT_X << endl;
      exit(1);
    } ENDTRY

    } // Monitor

  }

 cout << "CLIENT exits now ! " << endl;
 return 0;


} // main
```

Appendix C: Examples

Example 1

C.1 SERVERout1

```
> SERVER3 foxtail.dstc.edu.au contractA < SERVERInputA


[SERVER3: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SERVER3: Server "SERVER3" is now available to the network ]
[     Configuration tcp/1268/xdr ]
[SERVER3: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised)
]
contract type: contractA has 1 base interface(s):

contract

Interface type *contract*  has 6 contract elements:


element No. 0 is of type string with name partyA
Please enter the value for this element:

element No. 1 is of type string with name partyB
Please enter the value for this element:

element No. 2 is of type float with name partyA_gives
Please enter the value for this element:

element No. 3 is of type float with name partyB_gives
Please enter the value for this element:

element No. 4 is of type string with name date_of_agreement
Please enter the value for this element:

element No. 5 is of type string with name domain_name
Please enter the value for this element:

Interface type *contractA*  has 1 contract elements:


element No. 0 is of type string with name settlement_date
Please enter the value for this element:


The contract instance derived has the following form:
-----------------------------------------------------------------
El.No         Type          Name                        Value
-----------------------------------------------------------------
  1           string        partyA                      SP1
  2           string        partyB                      User
  3           float         partyA_gives                10
  4           float         partyB_gives                100
  5           string        date_of_agreement           16-July-1995
  6           string        domain_name                 ASC
  7           string        settlement_date             31-Aug-95
-----------------------------------------------------------------
Please enter value for maximum QoS that can be provided:
Please enter value for minimum price that can be accepted:
```

```
 QoSmax = 1.2  pricemin =0.8


Initial SERVER's Contract Instance created !
[SERVER3: New Connection
(foxtail.dstc.edu.au,23682,*,zoran,pid=23682,optimised) ]
CLIENT accepted: SERVER to start executing service
CurrentQoS variable may need to be checked during service execution !
[SERVER3: End of Connection (foxtail.dstc.edu.au,23682,*,,zoran,pid=23682) ]
```

C.2 CLIENTout1

```
> CLIENT1 foxtail.dstc.edu.au easy_going

Binding to the SERVER ...
[23682: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[23682: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=23656,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contractA
Initial offer is of the following form:
-----------------------------------------------------------------
El.No.         Type                          Name       Value
-----------------------------------------------------------------
  1            string                     partyA       SP1
  2            string                     partyB       User
  3            float                partyA_gives       10
  4            float                partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string            settlement_date       31-Aug-95
-----------------------------------------------------------------

Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[23682: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=23672,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
Contract is valid and SERVER is OK (reputable)

Next step is storing contract instance into Notary
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Notary now  ...
[23682: New Connection
(foxtail.dstc.edu.au,Notary,*,zoran,pid=23637,optimised) ]
The stored contract instance has number: 0

SERVER can now be invoked to start a service
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
n
CLIENT exits now !
```

C.3 CVout1

```
> CV2 foxtail.dstc.edu.au &


[1] 23672
[CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1269/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,23682,*,zoran,pid=23682,optimised)
]
Entered *checkValidity* operation
contract_type is: contractA
contract type: contractA has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractA*  has 1 contract elements:

Exit *checkValidity* operation
Enter *checkReputation* operation
Server_name SERVER3 answer = 1Exit *checkReputation* operation
[CV2: End of Connection (foxtail.dstc.edu.au,23682,*,,zoran,pid=23682) ]
```

C.4 Notaryout1

```
> Notary foxtail.dstc.edu.au &


[1] 23637
> [Notary: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[Notary: Server "Notary" is now available to the network ]
[     Configuration tcp/1267/xdr ]
Notary Component created !
[Notary: New Connection
(foxtail.dstc.edu.au,23682,*,zoran,pid=23682,optimised) ]
Entered *store* operation
Exit *store* operation
[Notary: End of Connection (foxtail.dstc.edu.au,23682,*,,zoran,pid=23682) ]
```

Example 2

C.5 CLIENTout2

```
CLIENT1 foxtail.dstc.edu.au negotiator


Binding to the SERVER ...
[26263: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[26263: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26142,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contract
Initial offer is of the following form:
-----------------------------------------------------------------------
El.No.         Type                          Name         Value
-----------------------------------------------------------------------
  1            string                        partyA       SP1
  2            string                        partyB       User
  3            float                  partyA_gives       10
  4            float                  partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
-----------------------------------------------------------------------

Next step is Negotiation (if client is negotiator)
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
CLIENT is: negotiator
Is CN to be used (for price)?
n
Negotiation between CLIENT and SERVER starts now

Enter Client's counteroffer
Enter a number for minimum acceptable QoS:
12
Enter a number for maximum price:
70
Client is sending counteroffer now ...
Client waits for SERVER's decesion ...
Server has not accepted the counteroffer - left to CLIENT to proceed
Enter Client's counteroffer
Enter a number for minimum acceptable QoS:
12
Enter a number for maximum price:
90  75
Client is sending counteroffer now ...
Client waits for SERVER's decesion ...
Server has not accepted the counteroffer - left to CLIENT to proceed
Enter Client's counteroffer
Enter a number for minimum acceptable QoS:
11 2
Enter a number for maximum price:
```

```
82
Client is sending counteroffer now ...
Client waits for SERVER's decesion ...
Server accepted counteroffer
Negotiation completed !

Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[26263: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26254,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
Contract is valid and SERVER is OK (reputable)

Next step is storing contract instance into Notary
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
n

SERVER can now be invoked to start a service
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
n
CLIENT exits now !
```

C.6 SERVERout2

```
SERVER3 foxtail.dstc.edu.au contract < SERVERInput


[SERVER3: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SERVER3: Server "SERVER3" is now available to the network ]
[     Configuration tcp/1277/xdr ]
[SERVER3: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised)
]


Interface type *contract*  has 6 contract elements:


element No. 0 is of type string with name partyA
Please enter the value for this element:

element No. 1 is of type string with name partyB
Please enter the value for this element:

element No. 2 is of type float with name partyA_gives
Please enter the value for this element:

element No. 3 is of type float with name partyB_gives
Please enter the value for this element:

element No. 4 is of type string with name date_of_agreement
Please enter the value for this element:

element No. 5 is of type string with name domain_name
Please enter the value for this element:


The contract instance derived has the following form:
-----------------------------------------------------------------------
El.No.         Type                                    Name
Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                 partyA_gives       10
  4            float                 partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
-----------------------------------------------------------------------
Please enter value for maximum QoS that can be provided:
Please enter value for minimum price that can be accepted:
 QoSmax = 1.2  pricemin =0.8


Initial SERVER's Contract Instance created !
[SERVER3: New Connection
(foxtail.dstc.edu.au,26263,*,zoran,pid=26263,optimised) ]
SERVER reads counter-offer and finds that it is:
-------------------------------------------------------
El.No.         Type                                Name
```

```
Value
------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                  partyA_gives       12
  4            float                  partyB_gives       70
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
------------------------------------------------------------

SERVER makes decesion whether to accept counter-offer or not ..

Since SERVER accepts min. price of 80 and can provide max. QoS level of 12
... and since Client's QoS offer is  12 and price offer is 70
SERVER not accepted-> up to Client to proceed ...
SERVER reads counter-offer and finds that it is:
------------------------------------------------------------
El.No.         Type                                  Name
Value
------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                  partyA_gives       12
  4            float                  partyB_gives       75
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
------------------------------------------------------------

SERVER makes decesion whether to accept counter-offer or not ..

Since SERVER accepts min. price of 80 and can provide max. QoS level of 12
... and since Client's QoS offer is  12 and price offer is 75
SERVER not accepted-> up to Client to proceed ...
SERVER reads counter-offer and finds that it is:
------------------------------------------------------------
El.No.         Type                                  Name
Value
------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                  partyA_gives       12
  4            float                  partyB_gives       82
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
------------------------------------------------------------

SERVER makes decesion whether to accept counter-offer or not ..

Since SERVER accepts min. price of 80 and can provide max. QoS level of 12
... and since Client's QoS offer is  12 and price offer is 82
SERVER will accept  and will start executing service
[SERVER3: End of Connection (foxtail.dstc.edu.au,26263,*,,zoran,pid=26263) ]
```

C.7 CVout2

```
> CV2 foxtail.dstc.edu.au &


[1] 26254
 [CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1278/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,26263,*,zoran,pid=26263,optimised)
]
Entered *checkValidity* operation
contract_type is: contract


Interface type *contract*  has 6 contract elements:

Exit *checkValidity* operation

Enter *checkReputation* operation
Server_name is:  SERVER3
This server has good record
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,26263,*,,zoran,pid=26263) ]
```

Example 3

C.8 CLIENTout3

```
> CLIENT1 foxtail.dstc.edu.au negotiator


Binding to the SERVER ...
[26496: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[26496: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26489,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contractB
Initial offer is of the following form:
-----------------------------------------------------------------------
El.No.         Type                            Name       Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                  partyA_gives       10
  4            float                  partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------------------

Next step is Negotiation (if client is negotiator)
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
CLIENT is: negotiator
Is CN to be used (for price)?
y
invoke CN here
Binding to the CN ...
[26496: New Connection (foxtail.dstc.edu.au,CN,*,zoran,pid=26485,optimised)
]
Client bound to CN
Best deal found through Negotiator is:
-----------------------------------------------------------------------
El.No.         Type                            Name         Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                  partyA_gives       12
  4            float                  partyB_gives       80
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------------------
Negotiation completed !
```

```
Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[26496: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26480,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
Contract is valid and SERVER is OK (reputable)

Next step is storing contract instance into Notary
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
n

SERVER can now be invoked to start a service
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
n
CLIENT exits now !
```

C.9 CNout3

```
> CN foxtail.dstc.edu.au &


[1] 26485
> [CN: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CN: Server "CN" is now available to the network ]
[     Configuration tcp/1241/xdr ]
[CN: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26489,optimised) ]
successfully bound to the SERVER
CN Component created !
[CN: New Connection (foxtail.dstc.edu.au,26496,*,zoran,pid=26496,optimised)
]

Entered BestDeal procedure

contract_type is contractB
 Initial offer is of the following form
-----------------------------------------------------------
El.No.         Type                                  Name
Value
-----------------------------------------------------------
  1            string                        partyA       SP1
  2            string                        partyB       User
  3            float                  partyA_gives       10
  4            float                  partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------
QoSfactor is: 1.2  pricefactor is: 0.8
SERVER can provide QoSmax of: 12 and will accept minimum of: 80 $
CN is sending counteroffer now ...
Server accepted counteroffer
Exit BestDeal procedure

[CN: End of Connection (foxtail.dstc.edu.au,26496,*,,zoran,pid=26496) ]
```

C.10 SERVERout3

```
 > SERVER3 foxtail.dstc.edu.au contractB < SERVERInputB &

[1] 26489
> [SERVER3: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SERVER3: Server "SERVER3" is now available to the network ]
[     Configuration tcp/1290/xdr ]
[SERVER3: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised)
]
contract type: contractB has 1 base interface(s):

contract

Interface type *contract*  has 6 contract elements:

element No. 0 is of type string with name partyA
Please enter the value for this element:

element No. 1 is of type string with name partyB
Please enter the value for this element:

element No. 2 is of type float with name partyA_gives
Please enter the value for this element:

element No. 3 is of type float with name partyB_gives
Please enter the value for this element:

element No. 4 is of type string with name date_of_agreement
Please enter the value for this element:

element No. 5 is of type string with name domain_name
Please enter the value for this element:

Interface type *contractB*  has 2 contract elements:


element No. 0 is of type string with name partyA_agentname
Please enter the value for this element:

element No. 1 is of type string with name partyB_agentname
Please enter the value for this element:


The contract instance derived has the following form:
-----------------------------------------------------------------------
El.No.         Type                        Name          Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                 partyA_gives        10
  4            float                 partyB_gives        100
  5            string          date_of_agreement         16-July-1995
  6            string                domain_name         ASC
  7            string           partyA_agentname         Agent1
  8            string           partyB_agentname         Agent2
-----------------------------------------------------------------------
```

```
Please enter value for maximum QoS that can be provided:
Please enter value for minimum price that can be accepted:
 QoSmax = 1.2  pricemin =0.8


Initial SERVER's Contract Instance created !
[SERVER3: New Connection
(foxtail.dstc.edu.au,CN,*,zoran,pid=26485,optimised) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,26496,*,zoran,pid=26496,optimised) ]
SERVER reads counter-offer and finds that it is:
----------------------------------------------------------------------
El.No.         Type                          Name       Value
----------------------------------------------------------------------
  1            string                     partyA       SP1
  2            string                     partyB       User
  3            float                partyA_gives       12
  4            float                partyB_gives       80
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string           partyA_agentname       Agent1
  8            string           partyB_agentname       Agent2
----------------------------------------------------------------------

SERVER makes decesion whether to accept counter-offer or not ..

Since SERVER accepts min. price of 80 and can provide max. QoS level of 12
... and since Client's QoS offer is  12 and price offer is 80
SERVER will accept  and will start executing service
[SERVER3: End of Connection (foxtail.dstc.edu.au,26496,*,,zoran,pid=26496) ]
```

C.11 CVout3

```
> CV2 foxtail.dstc.edu.au &


[1] 26480
> [CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1289/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,26496,*,zoran,pid=26496,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation

Enter *checkReputation* operation
Server_name is:  SERVER3
 This server has good record
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,26496,*,,zoran,pid=26496) ]
```

Example 4

C.12 CLIENTout4

```
> CLIENT1 foxtail.dstc.edu.au easy_going


Binding to the SERVER ...
[26784: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[26784: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26799,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contractB
Initial offer is of the following form:
-----------------------------------------------------------------------
El.No.         Type                          Name         Value
-----------------------------------------------------------------------
  1            string                        partyA       SP1
  2            string                        partyB       User
  3            float                  partyA_gives       10
  4            float                  partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------------------

Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[26784: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26805,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
Contract is valid and SERVER is OK (reputable)

Next step is storing contract instance into Notary
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Notary now  ...
[26784: New Connection
(foxtail.dstc.edu.au,Notary,*,zoran,pid=26801,optimised) ]
The stored contract instance has number: 0

SERVER can now be invoked to start a service
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
After what time [mins] do you want server to start
1
After what time [mins] do you want server to stop
5
SERVER will be started in 1 mins  and stopped after 5  mins
```

```
It is now Mon Aug  7 23:51:04 1995
Wait till SERVER starts ...
startT = 52
stopT = 57

 T->tm_min = 51 T->tm_sec = 4      startT = 52
 T->tm_min = 51 T->tm_sec = 14      startT = 52
 T->tm_min = 51 T->tm_sec = 24      startT = 52
 T->tm_min = 51 T->tm_sec = 34      startT = 52
 T->tm_min = 51 T->tm_sec = 44      startT = 52
 T->tm_min = 51 T->tm_sec = 54      startT = 52

Next step is reqesting actions of CM
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Monitor  now  ...
[26784: New Connection (foxtail.dstc.edu.au,CM,*,zoran,pid=26818,optimised)
]
succesfully bound to CM
 Pass arguments to Contract Monitor and fire it up!
What do you want CM to do? [record/invokeCE]
record
CLIENT exits now !
```

C.13CMout4

```
> CM foxtail.dstc.edu.au


[CM: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CM: Server "CM" is now available to the network ]
[     Configuration tcp/1258/xdr ]
CM Component created !
[CM: New Connection (foxtail.dstc.edu.au,26784,*,zoran,pid=26784,optimised)
]

Entered *request* operation

Contract number to be monitored is : 0
[CM: New Connection (foxtail.dstc.edu.au,Notary,*,zoran,pid=26801,optimised)
]
CM now bound to Notary to obtain Contract Instance
Mon Aug  7 23:52:11 1995

Contact Instance obtained from Notary is:
-----------------------------------------------------------------------
El.No.         Type                            Name       Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                 partyA_gives       10
  4            float                 partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string           partyA_agentname       Agent1
  8            string           partyB_agentname       Agent2
-----------------------------------------------------------------------
[CM: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26799,optimised) ]
 T->tm_min = 52 T->tm_sec = 11 stop = 57
 T->tm_min = 52 T->tm_sec = 31 stop = 57
 T->tm_min = 52 T->tm_sec = 51 stop = 57
Mon Aug  7 23:52:55 1995
 pCMlist->_length =0  value= 8
Mon Aug  7 23:52:57 1995
 pCMlist->_length =1  value= 8
 T->tm_min = 53 T->tm_sec = 11 stop = 57
 T->tm_min = 53 T->tm_sec = 31 stop = 57
Mon Aug  7 23:53:45 1995
 pCMlist->_length =2  value= 8
Mon Aug  7 23:53:47 1995
 pCMlist->_length =3  value= 8
 T->tm_min = 53 T->tm_sec = 51 stop = 57
 T->tm_min = 54 T->tm_sec = 11 stop = 57
 T->tm_min = 54 T->tm_sec = 31 stop = 57
Mon Aug  7 23:54:35 1995
 pCMlist->_length =4  value= 8
Mon Aug  7 23:54:37 1995
 pCMlist->_length =5  value= 8
 T->tm_min = 54 T->tm_sec = 51 stop = 57
 T->tm_min = 55 T->tm_sec = 11 stop = 57
```

```
Mon Aug  7 23:55:25 1995
 pCMlist->_length =6  value= 8
Mon Aug  7 23:55:27 1995
 pCMlist->_length =7  value= 8
 T->tm_min = 55 T->tm_sec = 31 stop = 57
 T->tm_min = 55 T->tm_sec = 51 stop = 57
 T->tm_min = 56 T->tm_sec = 11 stop = 57
Mon Aug  7 23:56:15 1995
 pCMlist->_length =8  value= 8
Mon Aug  7 23:56:17 1995
 pCMlist->_length =9  value= 8
 T->tm_min = 56 T->tm_sec = 31 stop = 57
 T->tm_min = 56 T->tm_sec = 51 stop = 57
Mon Aug  7 23:57:05 1995
 pCMlist->_length =10  value= 8
Mon Aug  7 23:57:07 1995
 pCMlist->_length =11  value= 8
Mon Aug  7 23:57:09 1995
 pCMlist->_length =12  value= 8
 T->tm_min = 57 T->tm_sec = 11 stop = 57
Monitor stops this monitoring

Exit *request* operation

[CM: End of Connection (foxtail.dstc.edu.au,26784,*,,zoran,pid=26784) ]
[CM: End of Connection (foxtail.dstc.edu.au,SERVER3,*,,zoran,pid=26799) ]
[CM: End of Connection (foxtail.dstc.edu.au,Notary,*,,zoran,pid=26801) ]
```

C.14SERVERout4

```
> SERVER3 foxtail.dstc.edu.au contractB < SERVERInputB


[SERVER3: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SERVER3: Server "SERVER3" is now available to the network ]
[     Configuration tcp/1255/xdr ]
[SERVER3: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised)
]
contract type: contractB has 1 base interface(s):

contract

Interface type *contract*  has 6 contract elements:

element No. 0 is of type string with name partyA
Please enter the value for this element:

element No. 1 is of type string with name partyB
Please enter the value for this element:

element No. 2 is of type float with name partyA_gives
Please enter the value for this element:

element No. 3 is of type float with name partyB_gives
Please enter the value for this element:

element No. 4 is of type string with name date_of_agreement
Please enter the value for this element:

element No. 5 is of type string with name domain_name
Please enter the value for this element:

Interface type *contractB*  has 2 contract elements:


element No. 0 is of type string with name partyA_agentname
Please enter the value for this element:

element No. 1 is of type string with name partyB_agentname
Please enter the value for this element:


The contract instance derived has the following form:
-----------------------------------------------------------------------
El.No.         Type                            Name       Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                 partyA_gives       10
  4            float                 partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------------------
```

```
Please enter value for maximum QoS that can be provided:
Please enter value for minimum price that can be accepted:
 QoSmax = 1.2  pricemin =0.8


Initial SERVER's Contract Instance created !
[SERVER3: New Connection
(foxtail.dstc.edu.au,SIM,*,zoran,pid=26827,optimised) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,26784,*,zoran,pid=26784,optimised) ]
CLIENT accepted: SERVER to start executing service
QoS variable may need to be checked during service execution !
pQoSlist->_length = 1
 It is now Mon Aug  7 23:51:04 1995

startT = 52
stopT = 57

 T->tm_min = 51 T->tm_sec = 4 startT = 52
 T->tm_min = 51 T->tm_sec = 14 startT = 52
 T->tm_min = 51 T->tm_sec = 24 startT = 52
 T->tm_min = 51 T->tm_sec = 34 startT = 52
 T->tm_min = 51 T->tm_sec = 44 startT = 52
 T->tm_min = 51 T->tm_sec = 54 startT = 52
Start producing QoS now ...
 try to bind to Sim ....
SERVER bound to sim
 send info to Sim
[SERVER3: End of Connection (foxtail.dstc.edu.au,26784,*,,zoran,pid=26784) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,CM,*,zoran,pid=26818,optimised) ]
```

C.15 Simout4

```
> SIM foxtail.dstc.edu.au < SimInput


[SIM: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SIM: Server "SIM" is now available to the network ]
[     Configuration tcp/1259/xdr ]
[SIM: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=26799,optimised) ]
Simulator object for SERVER's QoS variation  created !
Start producing QoS now ...
 try to bind to the SERVER ....
firstQoS = 10

T->tm_min = 52 T->tm_sec = 4 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 52 T->tm_sec = 54 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 53 T->tm_sec = 44 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 54 T->tm_sec = 34 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
```

```
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 55 T->tm_sec = 24 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 56 T->tm_sec = 14 stop = 57
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 57 T->tm_sec = 4 stop = 57
Will stop simulating service executiion now !!!
[SIM: End of Connection (foxtail.dstc.edu.au,SERVER3,*,,zoran,pid=26799) ]
```

C.16CVout4

```
> CV2 foxtail.dstc.edu.au


[CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1256/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,26784,*,zoran,pid=26784,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation

Enter *checkReputation* operation
Server_name is:  SERVER3
 This server has good record
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,26784,*,,zoran,pid=26784) ]
```

C.17Notaryout4

```
> Notary foxtail.dstc.edu.au


[Notary: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[Notary: Server "Notary" is now available to the network ]
[     Configuration tcp/1257/xdr ]
Notary Component created !
[Notary: New Connection
(foxtail.dstc.edu.au,26784,*,zoran,pid=26784,optimised) ]
Entered *store* operation
Exit *store* operation
[Notary: End of Connection (foxtail.dstc.edu.au,26784,*,,zoran,pid=26784) ]
[Notary: New Connection (foxtail.dstc.edu.au,CM,*,zoran,pid=26818,optimised)
]
Entered *retrieve* operation
Exit *retrieve* operation
```

Example 5

C.18 CLIENTout5

```
> CLIENT1 foxtail.dstc.edu.au easy_going


Binding to the SERVER ...
[27136: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[27136: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=27057,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contractB
Initial offer is of the following form:
-----------------------------------------------------------------------
El.No.         Type                          Name       Value
-----------------------------------------------------------------------
  1            string                      partyA       SP1
  2            string                      partyB       User
  3            float                 partyA_gives       10
  4            float                 partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string           partyA_agentname       Agent1
  8            string           partyB_agentname       Agent2
-----------------------------------------------------------------------

Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[27136: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26927,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
Contract is valid and SERVER is OK (reputable)

Next step is storing contract instance into Notary
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Notary now  ...
[27136: New Connection
(foxtail.dstc.edu.au,Notary,*,zoran,pid=27070,optimised) ]
The stored contract instance has number: 0

SERVER can now be invoked to start a service
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
After what time [mins] do you want server to start
1
After what time [mins] do you want server to stop
3
SERVER will be started in 1 mins  and stopped after 3  mins
```

```
It is now Tue Aug  8 01:00:58 1995
Wait till SERVER starts ...
startT = 1
stopT = 4

 T->tm_min = 0 T->tm_sec = 58      startT = 1

Next step is reqesting actions of CM
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Monitor  now  ...
[27136: New Connection (foxtail.dstc.edu.au,CM,*,zoran,pid=27089,optimised)
]
succesfully bound to CM
 Pass arguments to Contract Monitor and fire it up!
What do you want CM to do? [record/invokeCE]
invokeCE
CLIENT exits now !
```

C.19 CMout5

```
> CM foxtail.dstc.edu.au &


[1] 27089
 [CM: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CM: Server "CM" is now available to the network ]
[     Configuration tcp/1278/xdr ]
CM Component created !
[CM: New Connection (foxtail.dstc.edu.au,27136,*,zoran,pid=27136,optimised)
]

Entered *request* operation

Contract number to be monitored is : 0
[CM: New Connection (foxtail.dstc.edu.au,Notary,*,zoran,pid=27070,optimised)
]
CM now bound to Notary to obtain Contract Instance
Tue Aug  8 01:01:20 1995

Contact Instance obtained from Notary is:
------------------------------------------------------------------------
El.No.         Type                          Name       Value
------------------------------------------------------------------------
  1            string                      partyA       SP1
  2            string                      partyB       User
  3            float                partyA_gives       10
  4            float                partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string           partyA_agentname       Agent1
  8            string           partyB_agentname       Agent2
------------------------------------------------------------------------
[CM: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=27057,optimised) ]
 T->tm_min = 1 T->tm_sec = 20 stop = 4
 T->tm_min = 1 T->tm_sec = 40 stop = 4
Tue Aug  8 01:01:58 1995
will send message to CE
[CM: New Connection (foxtail.dstc.edu.au,CE,*,zoran,pid=27078,optimised) ]
 T->tm_min = 2 T->tm_sec = 1 stop = 4
Tue Aug  8 01:02:01 1995
will send message to CE
Tue Aug  8 01:02:03 1995
will send message to CE
 T->tm_min = 2 T->tm_sec = 21 stop = 4
 T->tm_min = 2 T->tm_sec = 41 stop = 4
Tue Aug  8 01:02:49 1995
will send message to CE
Tue Aug  8 01:02:51 1995
will send message to CE
Tue Aug  8 01:02:53 1995
will send message to CE
 T->tm_min = 3 T->tm_sec = 1 stop = 4
 T->tm_min = 3 T->tm_sec = 21 stop = 4
Tue Aug  8 01:03:39 1995
```

```
will send message to CE
 T->tm_min = 3 T->tm_sec = 41 stop = 4
Tue Aug  8 01:03:41 1995
will send message to CE
Tue Aug  8 01:03:43 1995
will send message to CE
 T->tm_min = 4 T->tm_sec = 1 stop = 4
Monitor stops this monitoring

Exit *request* operation

[CM: End of Connection (foxtail.dstc.edu.au,27136,*,,zoran,pid=27136) ]
```

C.20 CEout5

```
> CE foxtail.dstc.edu.au


[CE: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CE: Server "CE" is now available to the network ]
[     Configuration tcp/1276/xdr ]
CE Component created !
[CE: New Connection (foxtail.dstc.edu.au,CM,*,zoran,pid=27089,optimised) ]
[CE: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=27057,optimised) ]
CE sends message to SERVER ....
[CE: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26927,optimised) ]
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
CE sends message to SERVER ....
bound to CV
 add SERVER name to CV
```

C.21 CVout5

```
> CV2 foxtail.dstc.edu.au


[CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1277/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,27136,*,zoran,pid=27136,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation

Enter *checkReputation* operation
Server_name is:  SERVER3
 This server has good record
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,27136,*,,zoran,pid=27136) ]
[CV2: New Connection (foxtail.dstc.edu.au,CE,*,zoran,pid=27078,optimised) ]
Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
```

```
Exit *addServer* operation

[CV2: End of Connection (foxtail.dstc.edu.au,CE,*,,zoran,pid=27078) ]
[CV2: New Connection (foxtail.dstc.edu.au,27159,*,zoran,pid=27159,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation



Enter *checkReputation* operation
Server_name is:  SERVER3
 A non-performance has been previosly recorded for this server
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,27159,*,,zoran,pid=27159) ]
```

C.22Notaryout5

```
> Notary foxtail.dstc.edu.au


[Notary: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[Notary: Server "Notary" is now available to the network ]
[     Configuration tcp/1275/xdr ]
Notary Component created !
[Notary: New Connection
(foxtail.dstc.edu.au,27136,*,zoran,pid=27136,optimised) ]
Entered *store* operation
Exit *store* operation
[Notary: End of Connection (foxtail.dstc.edu.au,27136,*,,zoran,pid=27136) ]
[Notary: New Connection (foxtail.dstc.edu.au,CM,*,zoran,pid=27089,optimised)
]
Entered *retrieve* operation
Exit *retrieve* operation
```

C.23SERVERout5

```
> SERVER3 foxtail.dstc.edu.au contractB < SERVERInputB

[SERVER3: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SERVER3: Server "SERVER3" is now available to the network ]
[     Configuration tcp/1274/xdr ]
[SERVER3: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised)
]
contract type: contractB has 1 base interface(s):

contract

Interface type *contract*  has 6 contract elements:

element No. 0 is of type string with name partyA
Please enter the value for this element:

element No. 1 is of type string with name partyB
Please enter the value for this element:

element No. 2 is of type float with name partyA_gives
Please enter the value for this element:

element No. 3 is of type float with name partyB_gives
Please enter the value for this element:

element No. 4 is of type string with name date_of_agreement
Please enter the value for this element:

element No. 5 is of type string with name domain_name
Please enter the value for this element:

Interface type *contractB*  has 2 contract elements:


element No. 0 is of type string with name partyA_agentname
Please enter the value for this element:

element No. 1 is of type string with name partyB_agentname
Please enter the value for this element:


The contract instance derived has the following form:
-----------------------------------------------------------------------
El.No.         Type                                Name
Value
-----------------------------------------------------------------------
  1            string                       partyA       SP1
  2            string                       partyB       User
  3            float                 partyA_gives       10
  4            float                 partyB_gives       100
  5            string           date_of_agreement       16-July-1995
  6            string                 domain_name       ASC
  7            string            partyA_agentname       Agent1
  8            string            partyB_agentname       Agent2
-----------------------------------------------------------------------
```

```
Please enter value for maximum QoS that can be provided:
Please enter value for minimum price that can be accepted:
 QoSmax = 1.2  pricemin =0.8


Initial SERVER's Contract Instance created !
[SERVER3: New Connection
(foxtail.dstc.edu.au,SIM,*,zoran,pid=27094,optimised) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,27136,*,zoran,pid=27136,optimised) ]
CLIENT accepted: SERVER to start executing service
QoS variable may need to be checked during service execution !
pQoSlist->_length = 1
 It is now Tue Aug  8 01:00:58 1995

startT = 1
stopT = 4

 T->tm_min = 0 T->tm_sec = 58 startT = 1
Start producing QoS now ...
 try to bind to Sim ....
SERVER bound to sim
 send info to Sim
[SERVER3: End of Connection (foxtail.dstc.edu.au,27136,*,,zoran,pid=27136) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,CM,*,zoran,pid=27089,optimised) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,CE,*,zoran,pid=27078,optimised) ]
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
The following message arrived from Contract Enforcer
QoS degradation has been noted!
[SERVER3: End of Connection (foxtail.dstc.edu.au,SIM,*,,zoran,pid=27094) ]
[SERVER3: End of Connection (foxtail.dstc.edu.au,CE,*,,zoran,pid=27078) ]
[SERVER3: New Connection
(foxtail.dstc.edu.au,27159,*,zoran,pid=27159,optimised) ]
CLIENT accepted: SERVER to start executing service
QoS variable may need to be checked during service execution !
[SERVER3: End of Connection (foxtail.dstc.edu.au,27159,*,,zoran,pid=27159) ]
```

C.24SIMout5

```
> SIM foxtail.dstc.edu.au < SimInput


[SIM: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[SIM: Server "SIM" is now available to the network ]
[     Configuration tcp/1279/xdr ]
[SIM: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=27057,optimised) ]
Simulator object for SERVER's QoS variation  created !
Start producing QoS now ...
 try to bind to the SERVER ....
firstQoS = 10
 T->tm_min = 1 T->tm_sec = 8 stop = 4
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 1 T->tm_sec = 58 stop = 4
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 2 T->tm_sec = 48 stop = 4
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 3 T->tm_sec = 38 stop = 4
QoS delivery to be varied as follows:
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
```

```
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 10
writeQoS = 8
 T->tm_min = 4 T->tm_sec = 28 stop = 4
Will stop simulating service executiion now !!!
```

Example 6

C.25 CLIENTout6

```
> CLIENT1 foxtail.dstc.edu.au easy_going


Binding to the SERVER ...
[27159: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[27159: New Connection
(foxtail.dstc.edu.au,SERVER3,*,zoran,pid=27057,optimised) ]
Client bound to server

Next step is to check SERVER's initial offer
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Check SERVER's initial offer now ...

Contract type is: contractB
Initial offer is of the following form:
-----------------------------------------------------------------------
El.No.         Type                          Name       Value
-----------------------------------------------------------------------
  1            string                     partyA       SP1
  2            string                     partyB       User
  3            float                partyA_gives       10
  4            float                partyB_gives       100
  5            string          date_of_agreement       16-July-1995
  6            string                domain_name       ASC
  7            string           partyA_agentname       Agent1
  8            string           partyB_agentname       Agent2
-----------------------------------------------------------------------

Next step is Validation
DO YOU WANT TO CONTINUE WITH THIS ? [y/n]
y
Binding to the Contract Validator ...
[27159: New Connection (foxtail.dstc.edu.au,CV2,*,zoran,pid=26927,optimised)
]
sucessfully bound to CV
Contract is valid - check SERVER's reputation now
SERVER does not have a good reputation - not worth pursuing this further !!!
 Contract not valid ... will exit now
```

C.26CVout6

```
> CV2 foxtail.dstc.edu.au


[CV2: New Connection
(foxtail.dstc.edu.au,IT_daemon,*,orbix,pid=3836,optimised) ]
[CV2: Server "CV2" is now available to the network ]
[     Configuration tcp/1277/xdr ]
[CV2: New Connection (foxtail.dstc.edu.au,IR,*,zoran,pid=3946,optimised) ]
successfully bound to the IR
CV Component created !
[CV2: New Connection (foxtail.dstc.edu.au,27136,*,zoran,pid=27136,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation

Enter *checkReputation* operation
Server_name is:  SERVER3
 This server has good record
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,27136,*,,zoran,pid=27136) ]
[CV2: New Connection (foxtail.dstc.edu.au,CE,*,zoran,pid=27078,optimised) ]
Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
Exit *addServer* operation

Enter *addServer* operation
```

```
Exit *addServer* operation

[CV2: End of Connection (foxtail.dstc.edu.au,CE,*,,zoran,pid=27078) ]
[CV2: New Connection (foxtail.dstc.edu.au,27159,*,zoran,pid=27159,optimised)
]
Entered *checkValidity* operation
contract_type is: contractB
contract type: contractB has 1 base interface(s):
contract

Interface type *contract*  has 6 contract elements:


Interface type *contractB*  has 2 contract elements:

Exit *checkValidity* operation



Enter *checkReputation* operation
Server_name is:  SERVER3
 A non-performance has been previosly recorded for this server
Exit *checkReputation* operation

[CV2: End of Connection (foxtail.dstc.edu.au,27159,*,,zoran,pid=27159) ]
```

Bibliography

[1] N. Anderson. *Foundations of Information Integration Theory.* Academic Press, 1981.

[2] M.J. Aitken, A.W. Burrowes, R.D. Mulholland. *Investing on the Australian sharemarket*. Sun, Australia, 1994.

[3] Architecture Projects Management Ltd. (APM). *ANSAware 4.1, Application Programmer's Manual*. February 1993.

[4] K. Arrow, M. Intriligator. Agency and the Market. In, editors, K.Arrow, M. Intriligator, *Handbook of mathematical Economics, Vol. III*, pages 1183-1195, North-Holland, 1986.

[5] ASX Settlement and Transfer Corporation Pty. Ltd. *CHESS - An Overview,* Fifth Edition, November 1994.

[6] Australian Bureau of Transport and Communications Economics. *Quality of Service: Conceptual Issues and Telecommunications Case Study,* 1992.

[7] Australian Stock Exchange Ltd. *ASX Surveillance*.

[8] J.Bailey, L. McKnight. Internet Economics: What Happens When Constituencies Collide. In *Proceedings of 5th Annual Conference of the Internet Society, INET'95,* pages 659-666, Honolulu, Hawaii, USA, June 27-30, 1995.

[9] W. Barr, T. Boyd, Y. Inoue. The TINA Initiative. *IEEE Communications Magazine*, pages 70-76, March 1993.

[10] W. Barr, T. Boyd, M. Post. The Telecommunication Information Networking Architecture Initiative. In *Proceedings of the IFIP TC6/WG6.4 International Workshop on Open Distributed Processing*, Berlin, Germany, 8-11 October, 1991.

[11] M. Bearman. Trading in Open Distributed Environment (unpublished tutorial notes). *3rd IFIP TC6 International Conference on Open Distributed Processing*, February 1995, Brisbane, Australia.

[12] R. Benjamin, J. Blunt. Critical IT Issues: The Next Ten Years. *Sloan Management Review*, pages 7-19, Summer 1992, MIT.

[13] S. Berg, J. Jr. Lynch. The measurement and encouragement of telephone service quality. *Telecommunication Policy,* pages 210-224, April, 1992.

[14] T. Berners-Lee, et al. The World-Wide Web. *Communication of the ACM*, (37.8), pages 76-82, August 1994.

[15] T. Berners-Lee. The Evolution and Revolution of the Web. (unpublished Plenary Session notes). *The 5th Annual Conference of the Internet Society, INET'95*, Honolulu, Hawaii, USA, June 1995. (also available at: http://www.w3.org).

[16] A. Berry, K. Raymond. The *A1✓* Architecture Model. In, editors, K.Raymond, L.Armstrong, *Open Distributed Processing; Experiences with distributed environments, Proceedings of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 55-67, Chapman & Hall, 1995.

[17] A. Berry, K. Raymond. *The DSTC Architecture Model.* DSTC Internal Report, May 1994.

[18] G. Blair, T. Rodden. The Challenges of CSCW for Open Distributed Processing. In, editors J.D. Meer, B.Mahr, S.Strop, *Open Distributed Processing, Proceedings of the IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 127-145, North-Holland, 1994.

[19] R. Bons, R. Lee, R. Wagenaar, C. Wrigley. *Computer Aided Design of Interorganisational Trade Scenarios, A CASE for Open-edi*. Report. No. WP 94.03.01, Erasmus University Research Institute for Decision and Information Systems (EURIDIS), March 1994.

[20] H. Bowman, J. Derrick, M. Steen. Some Results on Cross Viewpoint Consistency Checking, In, editors, K.Raymond, L.Armstrong, *Open Distributed Processing; Experiences with distributed environments, Proceedings of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 399-412, Chapman & Hall, 1995.

[21] P. Boyle, J. Butterworth. The Principal/Agent Problem - Numerical Solutions. In *Economic Analysis of Information and Contracts,* pages 169-191, Essays in Honour of John Butterworth, 1988, Kluwer Academic Publishers, Boston.

[22] W. Brookes, D. Arnold, A. Berry, A. Bond, J. Indulska, K. Raymond. A Type Model Supporting Interoperability in Open Distributed Systems. In *TINA95 Conference Proceedings,* pages 275-191, February 95, Melbourne, Australia.

[23] P.G. Brown, P.V. Harrington. Defining Network Capabilities Using the Voice of the Customer. *IEEE Journal on Selected Areas in Communications*, Vol.12, No2, pages 228-233, February 1994.

[24] P. Brown. Digital Signatures: Are They Legal for Electronic Commerce? *IEEE Communications Magazine*, pages 76-80, September 1994.

[25] E. Brynjolfsson, H. Mendelson. Information Systems and the Organisation of Modern Enterprise. *Journal of Organisational Computing*, pages 245-255, Number 4, 1993.

[26] CCITT Blue Book. Terms and definitions related to the quality of telecommunication services, pages 257-267, Fascicle II.3, Rec. E.800, 1988.

[27] C. Chabernaud, S. Goerlinger. In, editor, P.W. Bayliss, Requirements on IN nodes to meet QoS objectives. In *Proceedings of the International Council for Computer Communications Intelligent Network Conference: Intelligent Networks, the path to global networking*, pages 51-62, Tampa, Florida, 1992.

[28] C. Ching, C. Holsapple, A. Whinston. Modelling Network Organisations: A Basis for Exploring Computer Support Coordination Possibilities. *Journal of organisational Computing, 3(3)*, pages 279-300, 1993.

[29] G. A. Jr. Churchill. *Marketing Research - Methodological foundations*. The Dryden Press, 1991.

[30] R. Cocchi. *Pricing in Multiple Service Class Computer Communication Network*. PhD Thesis, University of Southern California, December, 1992.

[31] R. Cohen. The Telecommunication Management Network (TMN). In, editor, Morris Sloman, *Network and Distributed Systems Management*, pages 217-243, Addison Wesley Publishing Company, 1994.

[32] K. Connolly et al. Partnering for Success: An Overview of Customer/Supplier Partnering. *IEEE Communications Magazine*, pages 46-51, October 1994.

[33] P. A. Cooper. Getting healthcare organisations to communicate effectively requires more than just EDI. In *Proceedings of the Second National Health Informatics Conference*, pages 13-20, Gold Coast, Australia, August 1-2 1994.

[34] G. Coulson. *Multimedia Application Support in Open Distributed Systems*. PhD thesis, Computing Department, Lancaster University, April 1993.

[35] G. Coulson, G. Blair. Meeting the real-time synchronisation requirements of multimedia in open distributed processing. In *Distributed Systems Engineering Journal 1(1994)*, pages 135-144.

[36] P. Crampton. The PCS Spectrum Auctions: Theory to Practice. In *Proceedings of the 3rd International Conference on Telecommunication Systems Modelling and Analysis*, pages 328-348, Nashville, USA, March 1995.

[37] R.M. Cyert, J. G. March. *A Behavioural Theory of the Firm*. Englewood Cliffs, (New Jersey), Prentice-Hall, 1963.

[38] J.M. Dalton. *How the stock market works*. New York Institute of Finance, 1988.

[39] T. Davenport, J. Short. The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, pages 11-27, Summer 1990, MIT.

[40] S. Douma, H. Schreuder. *Economic Approaches to Organisations*. Prentice Hall,'92.

[41] F. Dubinskas. Virtual Organisations: Computer Conferencing and Organisational Design. *Journal of Organisational Computing*, pages 389-416 3(4), 1993.

[42] A. Economides, J. Silvester. Multi-Objective Routing in Integrated Services Networks: A Game Theory Approach. In *Proceedings of INFOCOM'91*, pages 1220-1227, Bel Harbour, Florida.

[43] K.M. Eisenhardt. Agency Theory: An Assessment and Review. *Academy and Management Review*, pages 57-74 Vol. 14, No.1, 1989.

[44] The Encyclopedia Britannica, XVth Ed., Helen Hemingway Benton Publisher, Chicago, 1982, Vol. XVI, pages 904-907.

[45] The Encyclopedia Britannica, XVth Ed., Helen Hemingway Benton Publisher, Chicago, 1982, Vol. V, pages 124-127.

[46] E.F. Fama, M.C Jensen. Separation of ownership and control. *Journal of Law and Economics*, pages 301-26 vol. 26.S, 1983.

[47] D.F Ferguson. *The Application Of Microeconomics To The Design Of Resource Allocation And Control Algorithms*. PhD Thesis, Columbia University, 1989.

[48] FIRST VIRTUAL Holdings Incorporated. *FIRST VIRTUAL Internet Payment System*, available at http://www.fv.com.

[49] G. R. Gunther, N. Viswanthan. *The Human Edge: Information Technology and Helping People*, The Haworth Press, Inc., New York 1986.

[50] R. Gibbens, F. Kelly, G. Cope, M. Whitehead. Coalitions in International Network. In *Proceedings of the 7th UK Teletraffic Symposium*, pages 1-15, Durham University, April 1990.

[51] B.A. Godfrey, A.C. Endres. The Evolution of Quality Management Within Telecommunications. *IEEE Communications Magazine*, pages 26-35, October 1994.

[52] V. Gurbaxani, S. Whang S. The Impact of Information Systems on Organisations and Markets. *Communications of the ACM*, pages 60-73, January 1991.

[53] A. Halteren, P. Leydekkers, H. Korte. Specification and realisation of stream interface for the TINA-DPE. In *TINA95 Conference Proceedings*, pages 299-313, Melbourne, Australia, February 1995.

[54] R.J Harris. Concepts of Optimality in Alternate Routing Networks. *ATR. Vol. 7 No. 2*, pages 3-8, 1973.

[55] A. Herbert. The Challenges of Being Open (unpublished panel session notes). *The Fourth Telecommunications Information Networking Architecture Workshop*, L'Aquila, Italy, September 1993.

[56] B.R. Holstrom, J. Tirole. The Theory of the Firm. In, editors, R. Schmalensee, R.D. Willig, *Handbook of Industrial Organisation, Vol. I*, pages 63-133, 1989.

[57] W. Houser, J.A Griffin, C. Hage. *EDI Meets the Internet*. IETF-EDI WG, <draft-ietf-edi-faq-01.txt>, April 1995.

[58] M Hsiao, A. Lazar. Optimal decentralized flow control of Markovian queuing networks with multiple controller. *Performance Evaluation* 13, pages 181-204, 1991.

[59] B.A. Huberman, T. Hogg. The Emergence of Computational Ecologies. In, editors, L. Nadel, D. Stein, *SFI 1992 Lectures on Complex Systems*, pages 163-194, Addison-Wesley.

[60] D. Hutchison et al. *Quality of Service Management in Distributed Systems*, In, editor, M. Sloman, *Network and Distributed Systems Management*, pages 273-302, Addison-Wesley Publishing Company, 1994.

[61] D. Iggulden, O. Rees, R. van der Linden. *Architecture and Frameworks*, ANSA Phase III Technical Report, APM.1017.00.03 (Draft), June 1993.

[62] IONA Technologies Ltd. Orbix distributed object technology, Programmer's Guide, Release 1.3.1, February 1995.

[63] IONA Technologies Ltd. Orbix distributed object technology, Advanced Programmer's Guide, Release 1.3.1, February 1995.

[64] ISO/IEC CD-10746-1 ITU Recommendation X.901: *Open Distributed Processing - Reference Model - Part 1: Overview and guide to use*, July 1994.

[65] ISO/IEC 10746-2 ITU Recommendation X.902: *Open Distributed Processing - Reference Model - Part 2*, 1995.

[66] ISO/IEC IS 10746-3. *International Standard 10746-3, ITU-T Recommendation X.903: Open Distributed Processing - Reference Model - Part 3: Architecture*, January 1995.

[67] ISO/IEC CD 13235. *Committee Draft 13235: Open Distributed processing - Reference Model - Trading Function*, January 1995.

[68] ISO/IEC JTC 1/SC 21 N9309 *QoS - Basic Framework* - CD Text, January 1995.

[69] ISO/IEC JTC/WG3. *The Open-EDI Reference Model*. Working Draft document N255, 1994.

[70] M. Jensen, W. Meckling. Knowledge, Control and Organisational Structure: Parts I and II. In, editors, L. Werin, H. Hijkander, *Contract Economics*, pages 251-274, Cambridge, MA: Basil Blackwell, 1992.

[71] M. Katz. Vertical Contractual Relations. In, editors, R. Schmalensee, R.D. Willig, *Handbook of Industrial Organisation, Vol. I*, pages 655-720, 1989.

[72] B. Kelly. Becoming An Information Provider On The World Wide Web. In *Proceedings of 4th Annual Conference of the Internet Society, INET'945,* pages 122_1- 122_5.

[73] B. Kitson. CORBA and TINA: The Architectural Relationships. In *TINA95 Conference Proceedings*, pages 371-386, Melbourne, Australia, February 1995.

[74] J. Kramer. Distributed Systems. In editor M. Sloman, *Network and Distributed Systems Management,* pages 47-66, Addison-Wesley Publishing Company, 1994.

[75] D. M. Kreps. *Game Theory and Economic Modelling*. Oxford University Press, 1991.

[76] D.M. Kreps. *A Course in Microeconomic Theory.* Harvester Wheatsheaf, 1990.

[77] C. Lai, G. Medvinsky, B.C Neuman. Endorsements, Licensing, and Insurance for Distributed System Services. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 170-175, November 1994.

[78] K. Lancaster. *Consumer Demand. A New Approach*, Columbia University Press, 1971.

[79] A. Langsford. OSI Management Model and Standards. In, editor M. Sloman, *Network and Distributed Systems Management*, pages 69-93, Addison-Wesley Publishing Company, 1994.

[80] J.O. Ledyard. Incentive compatibility. *The New Palgrave dictionary of economics*, pages 739-743, 1986.

[81] R.M Lee, S.D Dewitz. Facilitating International Contracting: AI Extensions to EDI. University of Texas (also published in *International Information Systems*, January 1992).

[82] P. Linington. RM-ODP: The Architecture. In, editors, K.Raymond, L.Armstrong, *Open Distributed Processing; Experiences with distributed environments, Proceedings of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 15-33, Chapman & Hall, 1995.

[83] B. Lockwood. Pareto efficiency. In *The New Palgrave dictionary of economics*, pages 811-813, 1986.

[84] J.G.Jr. Lynch. Uniqueness Issues in the Decompositional Modelling of Multiattribute Overall Evaluations: An Information Integration Perspective. *Journal of Marketing Research,* pages 1-19, Vl. XXII (February 1985).

[85] J.G.Jr. Lynch, T. Buzas, S. Berg. Regulatory Measurement and Evaluation of Telephone Service Quality. *Management Science*, pages 169-193, Vol. 40, No.2, February 1994.

[86] J. MacKie-Mason, H. Varian. Pricing the Internet. University of Michigan, 1994.

[87] T. Malone, J. Yates, R. Benjamin. Electronic Markets and Electronic Hierarchies. *Communications of the ACM,* Vol. 30, No. 6, pages 484- 497, June 1987.

[88] T. Malone et al. Enterprise: a market-like task scheduller for distributed computing environments. In, editor, B.A. Huberman, *The Ecology of Computation*, pages 177-205, North-Holland, Amsterdam, 1988.

[89] J.G. March, H.A Simon. *Organisations*. New York: John Wiley, 1958.

[90] P.D.V Marsh. *Contract Negotiation Handbook,* 2nd Ed. Gower Publishing Company Limited, 1984.

[91] L. Mason. *Self-Optimizing Allocation Systems,* PhD Thesis, University of Saskatchewan, Aug. 1972.

[92] L. Mason, Z. Dziong, N. Tetreault. Fair-Efficient Call Admission Control Policies For Broadband Networks, In *Proceedings of the IEEE International Conference on Communications, ICC'93*, pages 976-982, May 1993, Geneva Switzerland.

[93] R. Mazumdar, L. Mason, C. Douligeris. Fairness in Network Optimal Flow Control: Optimality of Product Forms. *IEEE Transactions on Communications*, vol. pages 775-782, COM-39, May 1991.

[94] J.de Meer, A. Vogel. Quality of Service Workshop. In, editors, K.Raymond, L.Armstrong, *Open Distributed Processing; Experiences with distributed environments, Proceedings of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 511-512, Chapman & Hall, 1995.

[95] Z. Milosevic, A. Bond. Electronic Commerce on the Internet: What is Still Missing? In *Proceedings of the 5th Annual Conference of the Internet Society, INET'95,* pages 245-254, Honolulu, Hawaii, USA, June 1995.

[96] Z. Milosevic, A. Lister. New types of resource issues in open distributed systems: an agency theory modelling approach. In *Proceedings of the 3rd International Conference on Telecommunication Systems Modelling and Analysis*, pages 365-386, Nashville, USA, March 1995.

[97] Z. Milosevic, A. Lister, M. Bearman. New Economic-driven Aspects of the ODP Enterprise Specification and Related Quality of Service Issues. In, editors J.D. Meer, B.Mahr, S.Strop, *Open Distributed Processing, Proceedings of the IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 317-328, North-Holland, 1994.

[98] Z. Milosevic, Service provision and Quality of Service metrics: the TINA enterprise viewpoint. In *Proceedings of the Fourth Telecommunications Information Networking Architecture Workshop*, Vol. I, pages 189-204, L'Aquila, Italy, September 1993.

[99] Z. Milosevic, M. Phillips. Some New Performance Considerations in Open Distributed Environments. In *Proceedings of the IEEE International Conference on Communications, ICC'93*, pages 961-966, Geneva, May 1993.

[100] Z. Milosevic. *Open Distributed Systems: economic-driven issues*. Technical Report 290, Department of Computer Science, The University of Queensland, January 1994.

[101] Z. Milosevic, A. Berry, A. Bond, K. Raymond. Supporting Business Contracts in Open Distributed Systems. In *Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments (SDNE'95)*, pages 60-67, Whistler, Canada, IEEE Computer Society Press, June 1995.

[102] D. Minoli. *Broadband Network Analysis and Design*, Artech House, 1993.

[103] H. Mintzberg. *Mintzberg on Management*. New York: The Free Press, 1989.

[104] J.D. Moffett. Specification of Management Policies and Discretionary Access Control. In, editor, M. Sloman, *Network and Distributed Systems Management*, pages 455-479, Addison-Wesley Publishing Company, 1994.

[105] J.von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton: Princeton University Press, 1944.

[106] C. A. Nicolau. *A Distributed Architecture for Multimedia Communication Systems*. PhD thesis, University of Cambridge, May 1991.

[107] G. Nilsson, F. Dupuy, M. Chapman. An Overview of the Telecommunication Information Networking Architecture. In the *TINA95 Conference Proceedings,* pages 1-12, Melbourne, Australia, February 1995.

[108] Object Management Group. *Object Management Architecture Guide*, second edition, Sept. 1992. Revision 2.0, OMG TC Document 92.11.1.

[109] Object Management Group and X/Open. *The Common Object Request Broker: Architecture and Specification*, 1992.

[110] Object Management Group (Business Object Management Special Interest Group). *Description of a Business Object*, Version 2.0, Oct. 1994.

[111] Object Management Group. *Common Facilities Architecture*, Revision 4.0, OMG Document 95-1-1, January 1995.

[112] OECD Information Computer Communication Policy. *Performance indicators for public telecommunications operators*, 1990.

[113] A. Oodan. Quality of service and network performance. *British Telecommunication Journal*, pages 13.1.1- 13.1.8, issue 4/92.

[114] S. Oster. *Modern Competitive Analysis*. Oxford University Press, 1990.

[115] R. Pindyck, D. Rubinfield. *Microeconomics*. Macmillan Publishing Company, 1989.

[116] E. Rasmusen. *Games and information: an introduction to game theory.* Basil Blackwell, 1990

[117] K.A. Raymond. Reference Model of Open Distributed Processing: a Tutorial. In, editors J.D. Meer, B.Mahr, S.Strop, *Open Distributed Processing, Proceedings of the IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*. pages 3-14, North-Holland, 1994.

[118] K.A. Raymond. Reference Model of Open Distributed Processing: Introduction. In, editors, K.Raymond, L.Armstrong, *Open Distributed Processing; Experiences with distributed environments, Proceedings of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing*, pages 3-14, Chapman & Hall, 1995.

[119] R. Rees. The theory of principal and agent, Part1. *Bulletin of Economic Research* pages 3-26, 37:1, 1985.

[120] Reinecke, G.Dessler, W.F.Schoell. *Introduction to Business - a Contemporary View*, Allyn and Bacon, 1989.

[121] N.E. Renton. *Understanding the stock exchange*. The Investment Library, 1992.

[122] P. Resnick, R. Zeckhauser, C. Avery. Roles for Electronic Brokers. In, editor, Gerald Brock, *Toward a Competitive Telecommunication Industry: Selected Papers from the 1994 Telecommunications Policy Research Conference*, Chapter 15, Mahwah, NJ: Lawrence, Erlbaum Associates, 1995.

[123] J. S. Richters, C.A. Dvorak. Framework for defining quality of communications services. *IEEE Communications Magazine*, pages 17-23, October 1988.

[124] L.J Robison, P.J. Barry. *The Competitive Firm's Response to Risk*. Macmillan Publishing Comp., 1987.

[125] J. Rojot. *Negotiation: From Theory to Practice*. Macmillan, 1991.

[126] W. Rosenberg, D. Kenney D. *Understanding DCE*. Open System Foundation, 1992.

[127] A Rubinstein. *Game Theory in Economics*. The International Library of Critical Writings in Economics, Edward Elgar Publishing Ltd, 1990.

[128] J. Rymer. Business Objects. *Distributed Computing Monitor*, pages 3-31, Vol. 10, No.1, January 1995.

[129] B. Sanders. An Incentive Compatible Flow Control Algorithm for Rate Allocation in Computer Networks. *IEEE Transactions on Computers*, pages 1067-1072, Vol. 37, No. 9 Sept. 88.

[130] S. Schenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines, In *Proceedings of the SIGCOM'94 conference on Communications, Architecture, Protocols and Applications*, pages 47-57, August 1994.

[131] A. Schwartz. Legal Contract Theories and Incomplete Contracts. In, editors, L. Werin, H. Wijkander, *Contract Economics*, pages 76-109, Blackwell Publishers, 1992.

[132] R. Shelton. The Distributed Enterprise. *Distributed Computing Monitor,* pages 3-25, Vol.8, No. 10.

[133] M. Sloman, editor, *Network and Distributed Systems Management*, Addison Wesley Publishing Company, 1994.

[134] M. Sloman, K. Twidle. *Domains: A Framework for Structuring Management Policy.* In editor M. Sloman, *Network and Distributed Systems Management*, pages 433-453. Addison-Wesley Publishing Company, 1994.

[135] A. Smith. *The Wealth of Nations*, cannan edition, New York: Modern Library, 1937 (originally published 1776).

[136] R.M. Soley. Overview of OMG (unpublished tutorial notes). *The 3rd IFIP TC6 International Conference on Open Distributed Processing,* February 1995, Brisbane, Australia.

[137] K. Spremann. Agent and Principal. In, editors, G. Bamberg, K. Spermann, *Agency theory, information and incentives*, pages 3-37, Springer Verlag, 1989.

[138] S. Spriggs. Inside the Australian Stock Exchange. *Australian Communications,* pages 59-66, February 1995.

[139] J. Stiglitz. The Causes and Consequences of the Dependence of Quality on Price. *Journal of Economic Literature,* pages 1-48, March 1987.

[140] J. Stiglitz. Contract Theory and Macroeconomic Fluctuations. In, editors, L. Werin, H. Wijkander, *Contract Economics,* pages 292-323, Blackwell Publishers, 1992.

[141] J. Straub, R. Attner. *Introduction to Business*. PWS-KENT Publishing Company, Boston, 1988.

[142] A. Tanenbaum. *Computer Networks*. Prentice Hall, Inc., 1989.

[143] A. Tanenbaum. *Modern Operating Systems*. Prentice Hall, Inc., 1992.

[144] R.J. Tewels, E.S. Bradley. *The Stock Market*. John Wiley & Sons, 1987.

[145] J. Tirole. *The Theory of Industrial Organisation*. The MIT Press, Cambridge, Massachusetts, 1988.

[146] UN/EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport). *EDIFACT Syntax Rules* (ISO 9735) and *EDIFACT Data Element Directory* (ISO 7372), 1993.

[147] H. van der Heijden, R. Wagenaar. Value-added Information Services: Definition, Competitive Advantages, And Problems Of Agency. In *Proceedings of the 6th International Conference on Electronic Data Interchange*, Bled, Slovenia 1993.

[148] N. Venkatraman. IT-Enabled Business Transformation: From Automation to Business Scope Redefinition. *Sloan Management Review*, pages 73 -87, Winter 1994.

[149] C. Waldsburger et al. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, pages 103-117, February 1992.

[150] L. Werin, H. Wijkander. *Introduction*. In, editors, L. Werin, H. Wijkander, *Contract Economics*, pages 1-11, Blackwell Publishers, 1992.

[151] O. Williamson. Transaction-cost Economics: the Governance of Contractual Relation. In, editor, O. Williamson, *Industrial Organisation*, pages 223-251, The International Library of Critical Writings in Economics, 1990.

[152] S. Wolf et al. How Will We Rate Telecommunications System Performance. *IEEE Communications Magazine*, pages 23-29, October 1991.

[153] B. Wright. *The Law of Electronic Commerce, EDI, Fax and E-mail: Technology, Proof, and Liability.* Little Brown and Company, 1991.

[154] C.D. Wrigley, R.W Wagenaar, R.A Clarke. Electronic data interchange in international trade; frameworks for the strategic analysis of ocean port communities. In *Journal of Strategic Information Systems* 1994 3(3) pages 211-234.

[155] M. Young, Z. Milosevic. Open Distributed Systems and the Social Sciences: Focus on Welfare Service Professionals. In *Proceedings of the Third International Conference on Systems Integration, San Paulo*, Brazil, August 15-19, 1994.

[156] A. Zaheer, N. Venkatraman. Determinants of Electronic Integration in the Insurance Industry: An Empirical Test. *Management Science*, Vol. 40, No.5, pages 549-566, May 1994.